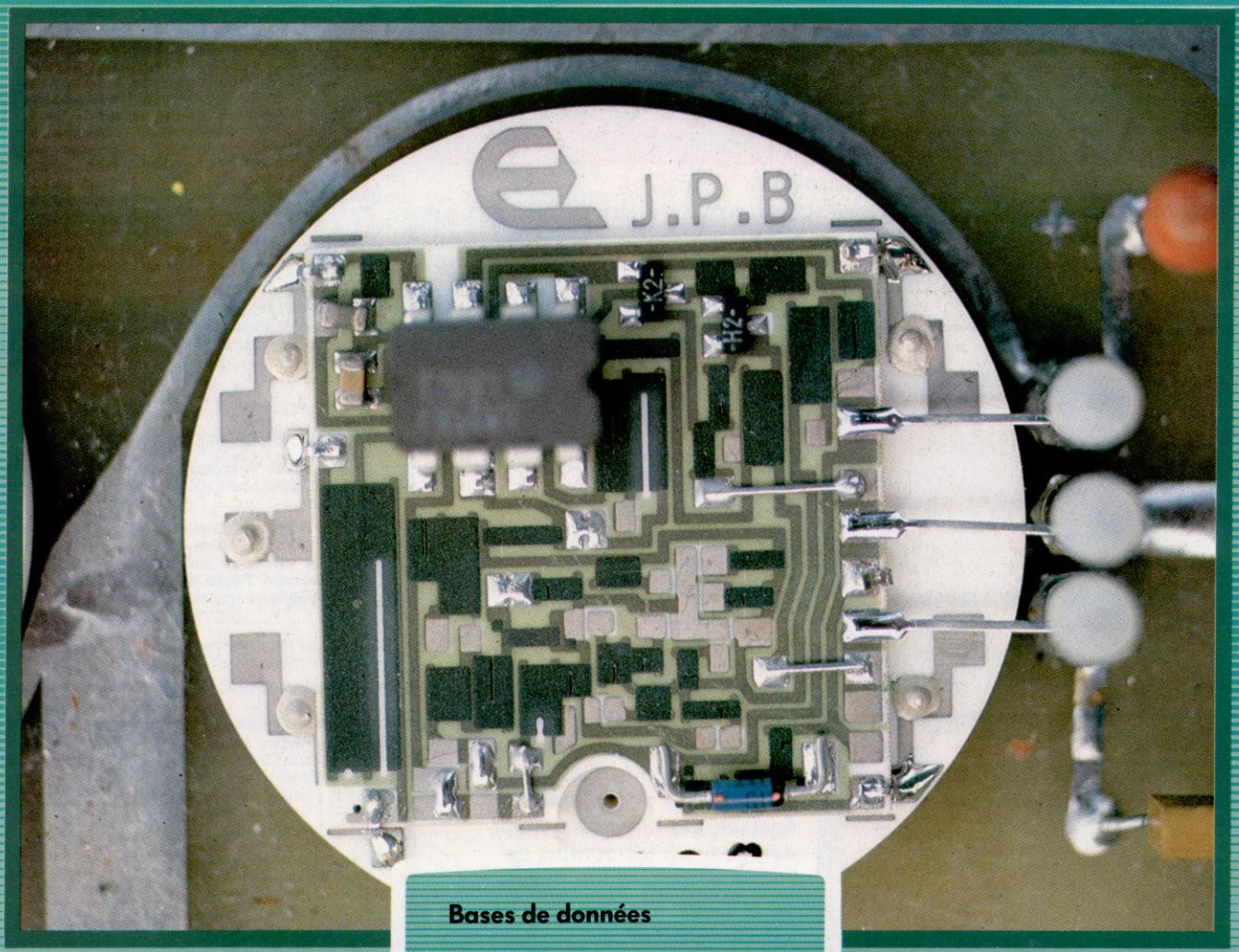
ABC INFORMATIQUE

N° 39

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Bases de données

Imprimante/Traceur

Mémoire de masse en cassette

L'Unité arithmétique et logique

EDITIONS ATLAS

M6062-39-12F 85FB-3,80FS-$1.95

Deux révolutions chez vous !

Prix : 175 FF

Découvrez le monde fascinant de la **RÉVOLUTION INFORMATIQUE** en devenant l'un des acteurs de **LA RÉVOLUTION FRANÇAISE**

La "Révolution française" est une disquette d'histoire publiée par les Éditions Atlas. Cette grande première bouleverse l'enseignement de l'histoire. Chaque utilisateur de la disquette "Révolution française" devient l'un des acteurs de l'événement et revit le déroulement implacable de la période révolutionnaire.

20 séquences • 74 dessins • 250 écrans graphiques

S'adresser aux services commerciaux des Éditions Atlas :
3, rue de la Taye, 28110 Lucé.
Tél. (37) 35.40.23.

LA RÉVOLUTION FRANÇAISE
Une révolution chaque fois recommencée
ÉDITIONS ATLAS

EDITIONS ATLAS • EDITIONS ATLAS • EDITIONS ATLAS • EDITIONS ATLAS •

Dans toutes les librairies

Les galaxies
nébuleuses, trous noirs, quasars

S'étendant sur près de 100 000 années-lumière, notre Galaxie comporte, outre ses 100 milliards d'étoiles, de la matière gazeuse, des poussières et des nébuleuses; divers indices semblent confirmer l'existence des fameux trous noirs. Mais notre Galaxie n'est pas la seule : on en dénombre des millions dans l'Univers. Synthèse de connaissances éparses, cet ouvrage permet de découvrir les confins de l'espace. Exploration fascinante.

Un volume relié sous jaquette illustrée. 160 pages. 68 photos, 49 dessins et schémas en couleurs. 65 photos en noir et blanc. Format : 22 × 28,5 cm.

EDITIONS ATLAS • EDITIONS ATLAS • EDITIONS ATLAS • EDITIONS ATLAS •

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 28-10-10. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS, Z.I. de Lucé, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume IV, n° 39.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Claire Bischoff *(correction)*.
Crédit photographique, couverture : Maillac-Réa.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : octobre 1984. 128410. Dépôt légal en Belgique : D/84/2783/27.

© Orbis Publishing Ltd., London.
© Éditions Atlas, Paris, 1984.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

A la base des données

Le type de logiciel le plus efficace pour la sauvegarde de données est la base de données. Nous allons voir quelles sont leurs limites sur des micros mais aussi la meilleure manière de les utiliser.

Une *base de données* est un ensemble d'informations se rapportant au même sujet. Ainsi, une base de données pourrait concerner les informations suivantes : les noms et adresses des membres d'un club, les frais des réunions du club, les dates et lieux des réunions, etc. La base de données pourra cependant concerner tous ces aspects à la fois, et les réunir en un seul grand fichier.

Un *fichier* est un ensemble d'enregistrements, chacun étant composé d'un certain nombre de zones. Pour certaines applications, comme les progiciels de comptabilité, la structure du fichier est déterminée par le logiciel. Cependant, pour une base de données, il est plus courant que la disposition du fichier soit à la discrétion de l'utilisateur afin de l'adapter à chaque application. Il convient donc de choisir la taille de chaque zone et son contenu. Pour un fichier d'adresses et de noms, les détails sur chaque personne occupent chacun un enregistrement. Le nom est sauvegardé dans la *zone de caractère,* et chaque ligne de l'adresse dans une zone particulière. Les zones ont habituellement 30 caractères de long, ce qui suppose au maximum 30 lettres dans chacune.

Outre les zones de caractères, il existe des *zones de nombres.* Elles permettent au programme d'effectuer des calculs sur les données ainsi stockées, et donc de produire de précieux rapports. Pour notre fichier des membres d'un club, le programme pourra calculer le nombre de membres ayant acquitté leur cotisation, les revenus courants pour l'année, et la somme à collecter. Mais toutes les données numériques ne doivent pas être stockées sur des zones numériques. Les numéros de téléphone sont l'exemple même de données sur lesquelles aucun calcul n'est à faire. Ce type de données figure aux rubriques de caractères.

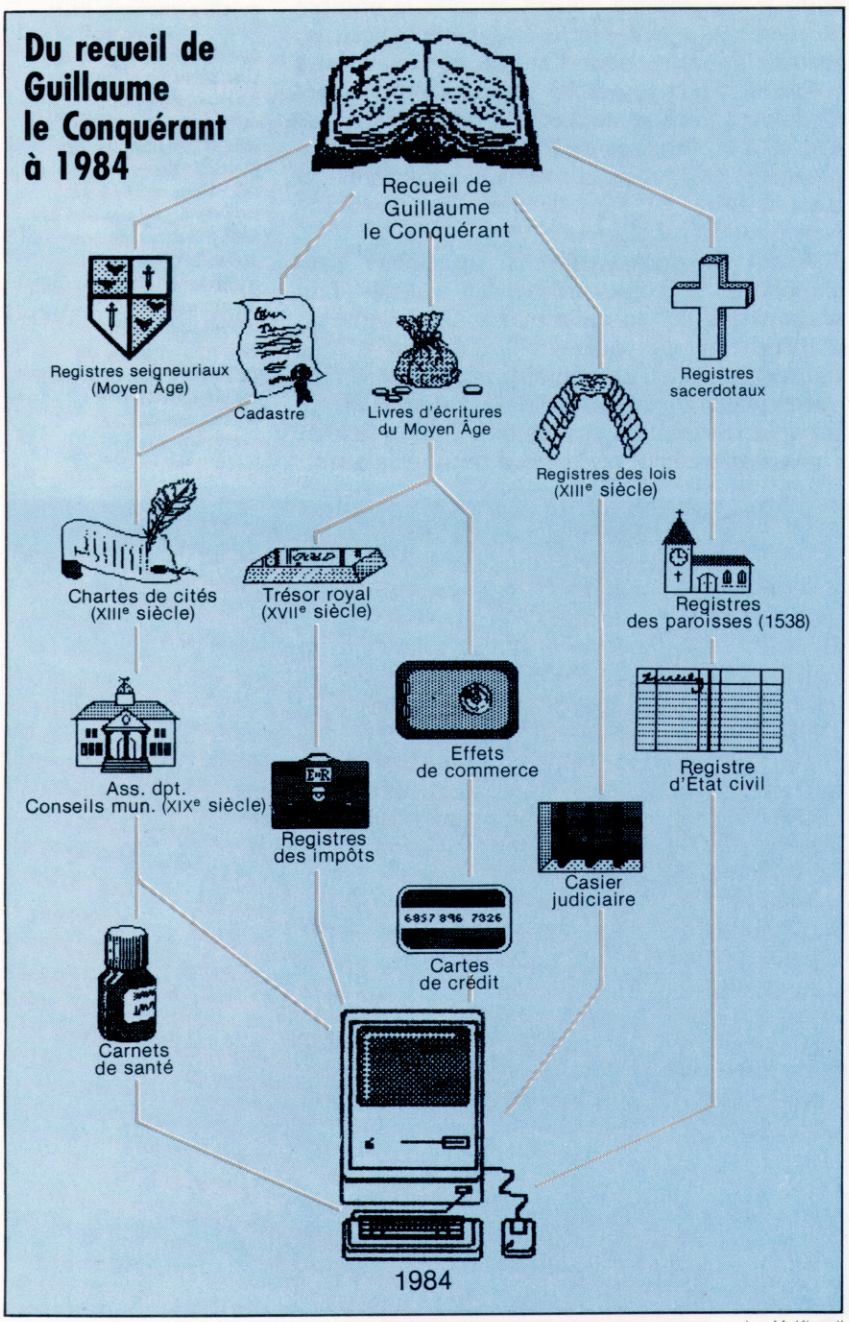
Avant de pouvoir dire si un fichier peut s'appliquer à votre ordinateur, il vous faut d'abord évaluer sa taille maximale (le nombre d'enregistrements que vous souhaitez y faire figurer), calculer la mémoire nécessaire à chaque enregistrement, et déterminer si vous disposez de suffisamment de mémoire pour le fichier. Un enregistrement composé d'un nom, de trois

Votre bloc-notes
La plupart des gens conservent des éléments d'information sur des sujets personnels sous forme de notes éparpillées sur de multiples feuilles. Une base de données sur micro-ordinateur peut être utilisée en regroupant les avis d'assurances, les titres de propriété, les relevés bancaires, etc. En outre, elle devient une aide inestimable pour tout collectionneur, qu'il s'agisse de timbres, de monnaies, d'enregistrements discographiques, ou de toute autre collection qui se prête à une classification systématique.
(Cl. Mike Brownlow.)

Banques de données
Dans les sociétés modernes, les banques de données sont nombreuses et facilement accessibles. Cette situation est le résultat d'une longue évolution. Les sociétés traditionnelles disposaient elles aussi de banques de données, mais elles étaient isolées les unes des autres par le temps et la distance. En revanche, l'interconnexion des banques de données informatisées, facilitera le travail des utilisateurs, mais se révélera un facteur menaçant pour les libertés individuelles, à l'instar du projet français SAFARI, aujourd'hui abandonné.

zones d'adresses de 30 caractères chacune et d'un numéro de téléphone de 10 caractères, occupe 130 octets. Si votre système de disque autorise 200 K par disque, vous pouvez alors loger 1 500 enregistrements par disquette.

Un système sur cassette, de 48 K, pourra probablement loger 300 enregistrements (laissant 10 K pour le programme et le système d'exploitation). En pratique, si vous voulez trier le fichier ou effectuer toute autre manipulation, vous aurez besoin de place mémoire de travail, et il est prudent de limiter la taille du fichier à la moitié de la place mémoire disponible. Les deux systèmes de stockage diffèrent totalement dans la mesure où la mémoire sur cassette doit être lue en mémoire et traitée comme un tout, alors que la rapidité d'accès du disque permet de laisser les fichiers sur le disque et d'être traités en mémoire.



Ian McKinnell

Un système de gestion de base de données vous permet de prendre une information à un fichier, les dépenses totales pour une année par exemple, et de la rapprocher d'une autre donnée en provenance d'un autre fichier, comme les revenus totaux des cotisations. Parmi les bases de données les plus simples, beaucoup d'entre elles ne gèrent qu'un seul fichier à la fois, de sorte que même si vous disposez des divers éléments d'information cités plus haut, vous ne pourrez les rapprocher comme avec une véritable base de données.

Un autre danger est de perdre accidentellement de précieuses données. Aussi est-il essentiel, pour des informations de grande importance, de garder toujours des copies de sauvegarde des fichiers et de faire des états imprimés réguliers de sorte que rien ne puisse être irrémédiablement perdu.

Une base de données sous sa forme la plus simple traitera un seul fichier. Le système se chargera de la saisie des données aussi facilement que pour remplir un questionnaire. Il vous permettra en outre d'avoir accès à l'information tant à l'écran que sur imprimante. L'utilisateur devra également être à même de pouvoir accéder à des enregistrements spécifiques par l'intermédiaire de critères de recherche et de tris du type « rechercher tous les membres du club n'ayant pas acquitté cette année leur cotisation et de même numéro de code postal ».

Un tel progiciel devra aussi pouvoir imprimer les zones voulues (le nom et la zone de l'adresse pour des mailings) et seulement celles-ci. Ce genre de base de données relativement simple existe pour presque tous les micros résidant sur cassette. Si vous disposez d'un système à disque, ces logiciels sont beaucoup plus performants.

Les progiciels les plus sophistiqués vous permettront d'entrer vous-même le format de vos données à l'écran afin de l'harmoniser avec des états préalablement imprimés. D'autres systèmes vous permettront de déterminer des seuils, ou nombre d'articles par exemple, devant être supérieurs ou inférieurs à un certain nombre. C'est pourquoi vous devrez soigneusement choisir votre base de données afin de trouver celle qui correspond le mieux à vos besoins.

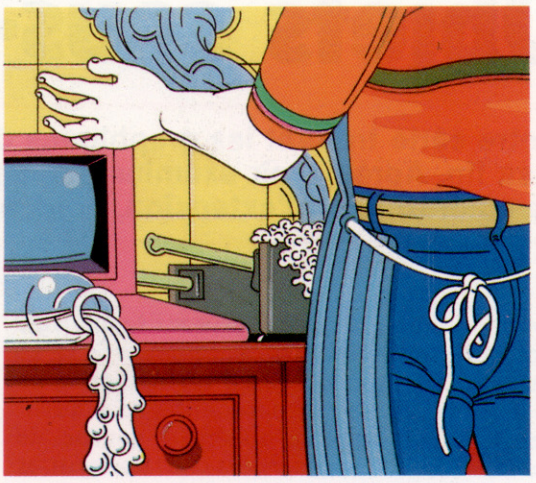
Un ordinateur équipé d'une base de données conviendra à des tâches répétitives ou à un type de traitement impliquant une grande quantité de données à trier ou à parcourir très rapidement. En revanche, certaines applications ne conviennent pas du tout.

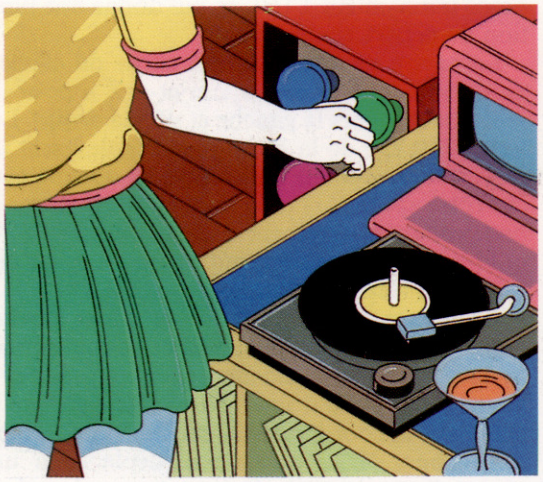
Il n'est pas approprié, sur un micro-ordinateur, d'effectuer, par exemple, un tri sur des numéros de téléphone à chaque fois que vous voulez téléphoner.

Le temps d'allumer votre ordinateur, de charger la base de données, d'appeler le fichier et d'effectuer le tri pour trouver le bon numéro de téléphone, prendre votre agenda et vous auriez pu passer cinq ou six coups de téléphone. Il se peut que vous vouliez faire figurer les noms et numéros de téléphone des personnes avec les-

Une recette catastrophique

C'est l'histoire édifiante d'Eric Personne qui est complètement subjugué par son micro, un Vic-20. Il faut reconnaître que ce dernier l'a familiarisé avec l'informatique et lui a permis d'apprendre un peu le Basic en le pratiquant. Le problème est qu'Eric ne peut pas « décrocher » de son ordinateur et confond applications sérieuses et amusement. C'est pourquoi il persiste à vouloir faire figurer le numéro de téléphone de ses (rares) amis sur ordinateur. Il utilise également une base de données pour ses recettes de cuisine. Aussi lorsque vous allez dîner chez lui, vous avez peu de chance de vous mettre à table avant minuit, car il s'obstine à vouloir suivre sur son ordinateur les recettes des plats qu'il prépare. Avec emphase, il le met en marche, s'efforce de charger le programme à partir de la cassette et patiente, radieux, le temps que l'ordinateur lise le fichier et cherche l'enregistrement. Éric déchiffre alors à l'écran la recette qui apparaît par petits groupes de lignes, avant d'aller finalement la mettre en pratique. Vous vous dépêchez d'arrêter l'ordinateur et de mettre la télévision. Et n'essayez pas d'arriver tard dans la soirée, pour échapper à cette mise en scène, il vous attendrait pour le cérémonial.





Roy Ingram

Les bons enregistrements

M^lle^ Tout-le-monde a toujours disposé d'une quantité énorme de disques. Elle fait parfois office de disc-jokey et son expérience lui a appris qu'il est vital de pouvoir mettre le bon disque au bon moment. Lucie (c'est son prénom), a décidé d'acquérir un micro-ordinateur. Pour des raisons de vitesse de traitement et de capacité de stockage, elle a choisi un système avec double lecteur de disques. Elle a ensuite entrepris une classification de ses disques pour construire sa base de données. Résultat : deux catégories principales de disques se subdivisaient elles-mêmes en sous-catégories, et ainsi de suite (structure arborescente). La distinction première était entre musique « blanche » et musique « noire ». Cette dernière regroupant reggae, jazz et soul. La musique « blanche » se divisait en rock et new wave. Lucie utilisait le système de classification de sa base de données pour sélectionner les disques qu'elle estimait convenir à une certaine atmosphère. Elle demandait par une commande, par exemple les disques de rock, un deuxième et peut-être aussi un troisième critères venant affiner la sélection. Ayant passé en revue la liste obtenue à l'écran, elle en faisait une impression.

quelles vous êtes en rapport, pour d'autres utilisations. Par exemple, écrire des mailings et produire des étiquettes, mais l'informatique ne saurait faire mieux, parfois, que les bons vieux systèmes manuels très simples.

Un système de base de données plus sophistiqué réunira tous les aspects précédemment abordés, et vous permettra d'effectuer en outre des opérations arithmétiques pour obtenir de nouvelles données ou de dégager des éléments significatifs. Plusieurs fichiers peuvent alors être mis en rapport, de sorte que les données s'appliquant à des domaines communs soient confrontées.

Ainsi, pour reprendre notre exemple du club, un même membre sera enregistré dans un premier fichier pour ce qui est de son affiliation, et dans un second fichier pour ses activités et performances. Lorsqu'un état récapitulatif le concernant sera nécessaire, les deux fichiers seront interrogés. Pour obtenir un instantané des performances sportives lors d'un tournoi de club, il suffira d'extraire tous les enregistrements à la date concernée. Cela évite de passer par tous les enregistrements pour une même personne. Afin de pouvoir exploiter un tel programme, un micro professionnel ou de gestion est nécessaire, avec un minimum de 64 K de mémoire et un lecteur de disques.

L'utilisation de systèmes de données s'est très largement généralisée ces dernières années. Les médecins les utilisent pour l'historique de leurs patients. Les chercheurs s'en servent pour effectuer des tris spécialisés et mettre au point des systèmes de référence et de renvois. Les gestionnaires les appliquent pour des mailings et l'envoi d'informations aux clients. Les bases de données ont alors pour rôle de tenir l'information à jour, outre celui de l'utiliser efficacement. Les bases de données peuvent maintenant traiter toutes formes de données, depuis de simples listes jusqu'aux systèmes complexes à plusieurs fichiers. Ceux-ci sont capables d'élaborer des études sophistiquées et de transmettre les données à d'autres logiciels tels que les traitements de texte. La possibilité de communiquer avec des bases de données publiques, sur le service Télétel, donne accès à des quantités énormes d'informations. Cela ouvre la voie à un proche futur où votre ordinateur personnel pourra faire office de terminal d'un système central.

Objets sous-marins

Notre série traitant les graphiques du Commodore 64 tire maintenant à sa fin. Après avoir examiné les mécanismes de conception, de coloration et d'extension, nous étudions les mouvements.

Pour positionner un plan objet sur l'écran, il est nécessaire de préciser des coordonnées X et Y. Chaque plan objet est défini sur une grille de 21 × 24 points, et les coordonnées sont mesurées à partir du coin supérieur gauche de l'écran. Elles sont mémorisées dans les registres de la puce VIC de la façon suivante :

Numéro du plan objet	0		1		2		3			7	
Coordonnée	X	Y	X	Y	X	Y	X	Y		X	Y
Registre VIC	V	V+1	V+2	V+3	V+4	V+5	V+6	V+7		V+14	V+15

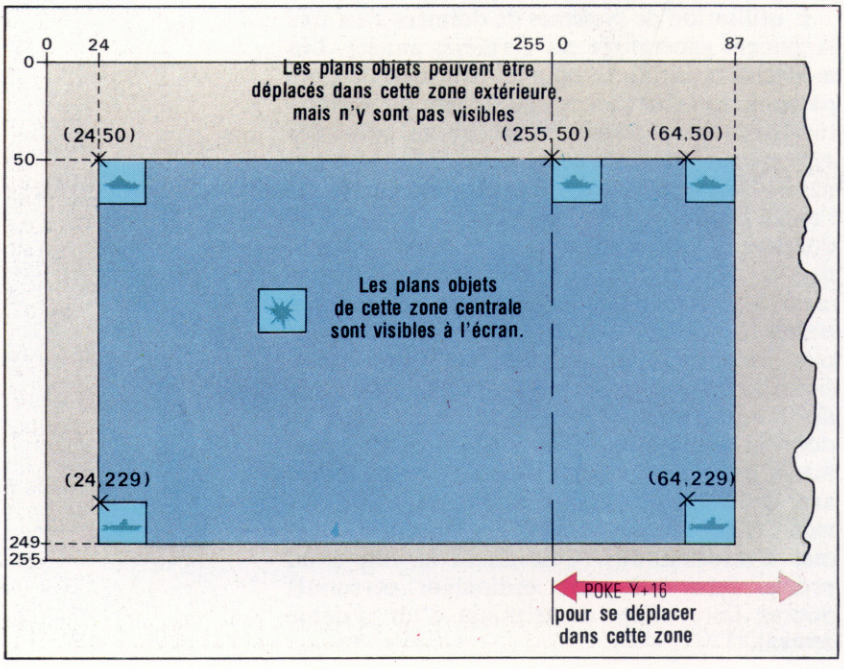
Chaque adresse peut accepter des nombres compris entre 0 et 255. C'est plus que suffisant pour déterminer l'un des 200 points dans la direction verticale (Y), et la capacité excédentaire sert à permettre la sortie d'un plan objet et son retour sur l'écran, au haut ou au bas de l'affichage. Cependant, les 320 points du plan horizontal excèdent le maximum de 255 permis par un registre à 8 bits, un bit supplémentaire est donc affecté à chaque plan objet. Ces bits supplémentaires sont groupés dans un registre unique à l'adresse V+16. Le diagramme suivant illustre les limites d'écran pour des plans objets non agrandis et entièrement visibles.

Le mouvement du navire est commandé par les lignes 230-250 et 270-290 du programme. Les coordonnées initiales du navire sont données à la ligne 2270 de la routine de création du plan objet. Le navire ne peut se déplacer que dans une direction horizontale, la coordonnée Y demeurera donc inchangée. Si celle-ci est placée à 80, le navire sera correctement positionné sur l'océan qui a été créé par la routine de préparation d'écran. La coordonnée X du navire, X0, est posée initialement à 160, ce qui donne une position de départ centrale.

Le navire est commandé à partir du clavier, au moyen des touches « Z » et « X » pour les mouvements vers la gauche et vers la droite. Les lignes 230-250 de la boucle principale lisent un caractère provenant du clavier. Si aucune touche n'est appuyée, l'exécution du programme se poursuit (contrairement à la commande INPUT qui interrompt le programme jusqu'à ce qu'une touche soit appuyée).

La valeur obtenue par GET est alors stockée dans A$ et utilisée pour modifier la coordonnée X du navire : si la touche « Z » est pressée, le navire se déplace d'une courte distance vers la gauche et la coordonnée X est diminuée ; si c'est la touche « X », le navire se déplace sur une courte distance vers la droite et la coordonnée X est augmentée. La seconde partie des lignes 240 et 250 renferme un test qui vérifie si le navire a atteint les limites de sa course. La structure IF...THEN du BASIC Commodore est telle que si la première condition d'une ligne de programme est fausse, le reste de la ligne n'est pas exécuté.

Aux lignes 300-350 de la boucle principale, la valeur de DX sélectionnée est ajoutée à la coordonnée X du sous-marin, et cette valeur est testée afin de s'assurer que le sous-marin n'a pas atteint le bord de l'écran. Les lignes 340 et 350 permettent de résoudre le problème posé par des valeurs de X3 situées au-delà de 255. Lorsque la coordonnée X atteint 256, deux choses doivent se produire — le bit correspondant au plan objet 3 dans le registre V+16 est mis à un et le registre normal de la coordonnée X est remis à zéro. Le tableau suivant illustre ce qui se produit dans les registres lorsque le sous-marin franchit la limite X = 255 :



La position du plan objet
Ce diagramme illustre les paramètres utilisés pour positionner un plan objet sur l'écran du Commodore 64. Lorsqu'il est placé dans la partie centrale, un plan objet est visible. Si le plan objet est déplacé vers la partie extérieure de l'affichage, il ne sera plus visible. Les quatre plans objets dans le coin de la zone centrale illustrent jusqu'où les plans objets peuvent être déplacés pour être toujours entièrement visibles. Les coordonnées du coin supérieur gauche du plan objet sont données. (Cl. Kevin Jones.)

X3	V+16								V+6							
254	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
255	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
256	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
257	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

La variable H3 sera mise à 1 si la valeur de X3 dépasse 255. De même, L3 sera remis à zéro si H3 devient 1. Les valeurs de L3 et de H3 peuvent alors être écrites dans les registres V+6 et V+16.

Lancement des charges explosives

Pendant le jeu, les charges explosives peuvent être lâchées à tout moment sur le sous-marin. Pour simplifier la programmation, nous devrons faire en sorte que lorsqu'une charge a été lancée, aucune autre ne pourra être lancée jusqu'à ce que :

a) le sous-marin ait été touché; ou

b) les charges aient raté le sous-marin et l'aient dépassé légèrement; ou

c) les charges aient raté le sous-marin et atteint le fond de la mer.

La boucle principale du programme effectue deux tâches par rapport aux charges explosives : elle doit détecter l'utilisation de la touche « M », et dès le lancement de la charge explosive, elle doit contrôler le mouvement vertical. Le programme doit aussi veiller à ce qu'aucune charge ne soit lancée lors de la descente d'une autre charge. Ce dernier problème peut être résolu grâce à l'emploi d'un drapeau. Il s'agit d'une technique souvent appliquée dans le contrôle d'un programme, afin de signaler si un événement s'est ou ne s'est pas produit. Dans notre programme, nous utiliserons la variable FL pour signaler la descente d'une charge explosive. Elle aura 1 comme valeur si une charge descend, sinon sa valeur sera zéro. A la ligne 100 du programme, la valeur de FL est mise à zéro. La ligne 260 appelle le sous-programme « Préparer les charges explosives » à la ligne 3000 si M est appuyé et si le drapeau est mis à zéro. Un second sous-programme est utilisé à la ligne 4000 pour déplacer une charge explosive lancée, et il est appelé à la ligne 360.

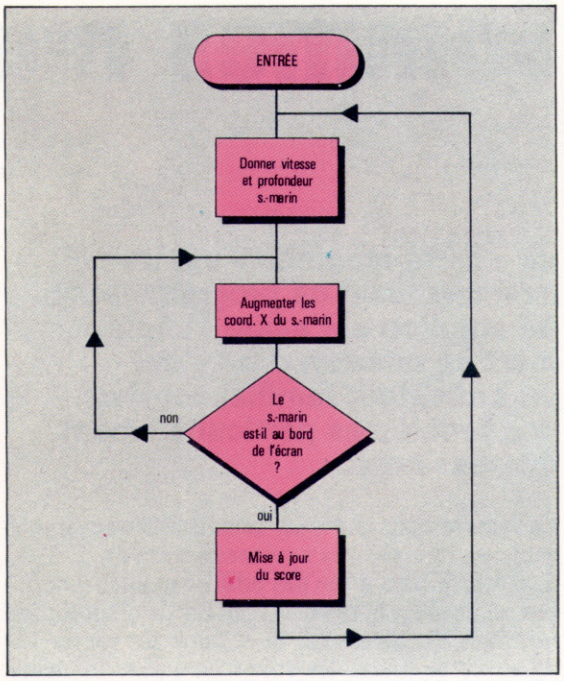
Le sous-programme « Préparer les charges explosives » a trois fonctions :

1. Mettre le drapeau FL à 1 pour signaler qu'une charge a été lancée.
2. Définir les coordonnées initiales : la coordonnée X est celle du navire et la coordonnée Y est initialement 95, juste sous la surface de la mer.
3. Illuminer le plan objet-charge explosive.

Le sous-programme « Déplacer la charge explosive » sert à déplacer la charge explosive vers le bas de l'écran. De plus, des tests doivent être faits pour voir si :

1. La charge explosive est passée devant le sous-marin ou a atteint le fond de la mer.
2. La charge explosive a touché le sous-marin.

Si le premier événement a eu lieu, le plan objet de la charge explosive peut disparaître et le drapeau peut être remis à zéro, ce qui autorise le lancement d'une autre charge explosive. Le second événement est testé en utilisant une autre caractéristique des plans objets du Commodore

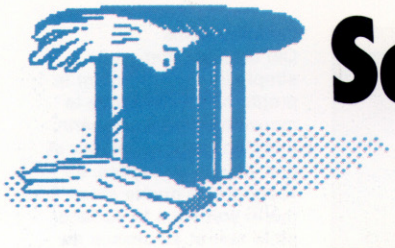


Organigramme
Cet organigramme très simple illustre comment le programme commande le mouvement du sous-marin. Il obtient une profondeur et une vitesse aléatoires, en veillant à ce que le sous-marin soit sous la surface de la mer et au-dessus du fond de la mer. Le sous-marin traverse alors la totalité de l'écran.
(Cl. Kevin Jones.)

64 — le registre de collision des plans objets. Comme d'autres registres de la puce VIC, ce registre, V+30, dispose d'un bit pour chaque plan objet. Si un plan objet est impliqué dans une collision avec un autre plan objet, les bits correspondants dans ce registre sont mis à un. Par conséquent, si le sous-marin (plan objet 3) et la charge explosive (plan objet 2) entrent en collision, le contenu du registre V+30 sera 12 (00001100). En lisant ce registre et en testant son contenu, nous pouvons savoir si la charge explosive a touché le sous-marin. Si c'est le cas, un autre sous-programme HIT (touche) est appelé à la ligne 5000. Ce sous-programme sera examiné dans la dernière partie de ce projet, avec les instructions servant à mettre à jour le POINTAGE MAXI (HI SCORE) et à redémarrer le jeu.

Sous-programmes de déplacement du sous-marin

```
130 GOSUB 2500 :REM DEF COORD SOUS-MARIN
230 GET A$
240 IF A$="Z" THEN XO=XO-1.5: IF XO 24
THEN XO=24
250 IF A$="X" THEN XO=XO+1.5: IF XO>240
THEN XO=245
260 IF A$="M" AND FL=0 THEN GOSUB 3000:
    REM PREPARER LES CHARGES EXPLOSIVES
265:
270 REM ** DEPLACER LE NAVIRE **
280 POKE V,XO
295
300 REM ** DEPLACER LE SOUS-MARIN **
310 X3=X3+DX
315 :
320 REM ** SOUS-MARIN AU BORD DE L'ECRAN? **
330 IF X3>360 THEN DS=-1:GOSUB 5500:
GOSUB 2500
340 113=INT(X3/256):L3=X3-256*113
350 POKE V+6,L3:POKE V+16,PEEK(V+16)OR(8*113)
360 IF FL=1 THEN GOSUB 4000:
    REM DEPLACER CHARGE EXPLOSIVE
370 GOTO 270 :REM REDEMARRER BOUCLE PRINCIPALE
380:
390:
2500 REM **** REDEFINIR LES COORD
     DU SOUS-MARIN **
2510 Y3=110+INT(RND(TI)*105)
2520 X3=0:DX=RND(TI)*3+1
2530 POKE V+7,Y3:POKE V+6,X3
2540 POKE V+16,PEEK(V+16)AND 247
2550 RETURN
2560:
2570:
3000 REM *** PREPARER LES CHARGES EXPLOSIVES **
3010 :
3020 REM ** DEF DRAPEAU **
3030 FL=1
3040 :
3050 REM ** DEF COORD **
3060 Y2=95:X2=X0
3070 POKE V+4,X2 : POKE V+5,Y2
3080 :
3090 REM ** ILLUMINER PLAN OBJET 2 **
3100 POKE V+21, PEEK (V+21)OR4
3110 RETURN
3120 :
3130 :
4000 REM ** DEPLACER CHARGE EXPLOSIVE **
4010 :
4020 REM ** AUGMENTER COORD Y **
4030 Y2=Y2+2
4040 POKE V+5,Y2
4050 :
4060 REM ** TEST BAS ET EFFACER **
4070 IF Y2>Y3+25 OR Y2 216 THEN
POKE V+21,PEEK(V+21) AND 251:FL=0
4080 :
4090 REM ** TEST POUR IMPACT SOUS-M **
4100 IF PEEK(V+30)=12 THEN GOSUB 5000:
REM SOUS-PO TOUCHE
4110 RETURN
4120 :
4130 :
```



Sommes magiques

Le « carré magique » est un exemple idéal de programmation. Sa solution est assez simple, mais sa mise en œuvre sur un ordinateur familial soulève des problèmes particulièrement intéressants...

Le « carré magique » est constitué d'une simple grille où figurent des nombres entiers (1, 2, 3, 4, 5, etc.); le zéro n'est pas admis, ni aucun nombre en double. L'objet du jeu est de disposer les nombres de sorte que la somme de toutes les lignes et de toutes les colonnes donne le même résultat. La plus simple des grilles possibles est de 3 fois 3 cases, avec la disposition suivante pour les nombres :

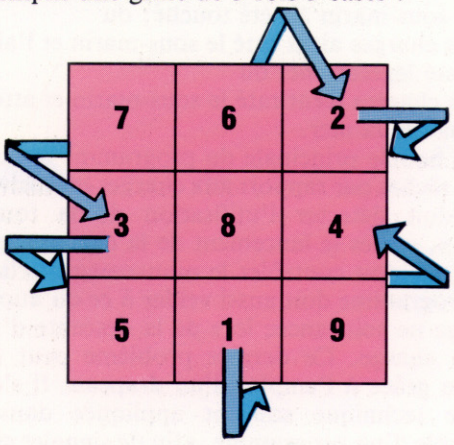
7	6	2	15
3	8	4	15
5	1	9	15
15	15	15	

Dans cet exemple, toutes les lignes et toutes les colonnes ont la même somme de 15. Comme exercice, essayez de remplir une grille de 5 fois 5 cases avec des nombres de 1 à 25. Cela n'est pas aussi facile qu'il n'y paraît.

Fort heureusement, votre ordinateur peut simplifier votre tâche. Vous pouvez par exemple lui demander, par un programme, de remplir la grille avec des nombres, et de tester les sommes de chaque ligne et de chaque colonne. Cette méthode présente l'inconvénient majeur de pouvoir prendre des heures avant de parvenir à une solution, et ce, même pour une petite grille de 7 ou 9 positions. Ce qu'il faut, c'est une méthode pour générer les nombres. Cela vous éviterait d'avoir à les trouver par vous même. Voici une méthode possible :

1. Commencez par mettre le chiffre 1 sur la position médiane de la dernière ligne.
2. Après avoir rempli une position, déplacez-vous sur la position immédiatement en dessous et à droite, à laquelle vous affecterez le nombre suivant (2). Si le déplacement vers la droite vous amène sur la dernière position à droite de la grille, allez directement à la première colonne.
3. Si la position suivante selon cet ordre est déja prise, allez à une position à gauche de la dernière occupée.
4. Continuez ainsi jusqu'à la dernière position libre.

En appliquant cette méthode, on voit comment se remplit une grille de 3 fois 3 cases :



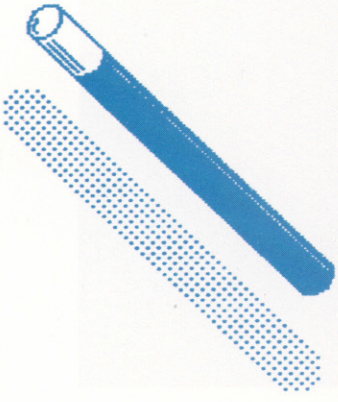
Il est facile d'écrire un programme qui suit cette démarche pour toute grille.

Vous devrez simplement créer pour cela un tableau à deux dimensions de la taille voulue, et utiliser ensuite une boucle pour le remplir selon les règles fournies plus haut. Vous remarquerez

Carrés magiques

Les carrés suivants ont été remplis par le programme et vérifiés par lui comme étant exacts, ainsi que l'indique le message. Pour la grille de 15 fois 15, il est possible de mettre en évidence un motif en diagonale composé de nombres à 3 et à 2 chiffres. C'est la conséquence de l'algorithme de remplissage.

52	42	32	22	12	2	73	72	62
63	53	43	33	23	13	3	74	64
65	55	54	44	34	24	14	4	75
76	66	56	46	45	35	25	15	5
6	77	67	57	47	37	36	26	16
17	7	78	68	58	48	38	28	27
19	18	8	79	69	59	49	39	29
30	20	10	9	80	70	60	50	40
41	31	21	11	1	81	71	61	51



en appliquant ce programme que cette méthode n'est valable que pour un nombre impair de positions. Votre programme devra refuser les grilles dont le nombre de positions est pair.

L'affichage de la grille devra être particulièrement soigné. Pour davantage de lisibilité, les nombres seront alignés en lignes et en colonnes. Cet alignement s'obtient facilement si votre micro comporte la commande PRINT USING. Si ce n'est pas le cas, vous devrez transcrire le nombre à imprimer, sous la forme d'une chaîne. Cette chaîne sera complétée par des caractères d'espacement, afin qu'elle ait toujours la même longueur. Voici un sous-programme qui assure cette transcription :

```
1000 REM TRANSCRIRE A EN A$ ET ALIGNER
1010 A$=STR$(A)
1020 IF LEN(A$)< 3THENA$=" "+A$:GOTO 1020
1030 RETURN
```

La méthode exacte dépendra bien sûr de votre ordinateur.

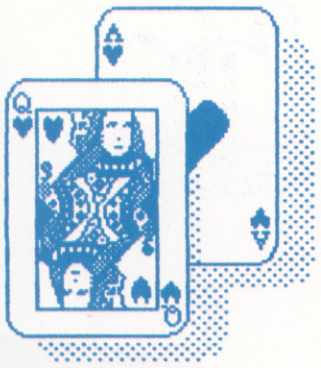
Le prochain problème est celui de la taille de l'écran. La plupart des micros ne sont pas capables de faire figurer à l'écran des grilles de grandes dimensions. Un écran de 40 colonnes peut recevoir 13 colonnes à 2 chiffres, mais une grille de 13 fois 13 cases comportera également des positions à 3 chiffres qui ne pourront donc pas être logées. Aussi le maximum possible sera de 9 fois 9. Une imprimante permettra de créer des grilles beaucoup plus grandes. La plupart des imprimantes peuvent imprimer 80 ou 132 colonnes, et des grilles encore plus grandes seront imprimables par parties.

Le but ultime du projet est de créer la grille la plus grande possible, et de la présenter de la manière la plus lisible.

130	114	98	82	66	50	34	18	2	211	210	194	178	162	146
147	131	115	99	83	67	51	35	19	3	212	196	195	179	163
164	148	132	116	100	84	68	52	36	20	4	213	197	181	180
166	165	149	133	117	101	85	69	53	37	21	5	214	198	182
183	167	151	150	134	118	102	86	70	54	38	22	6	215	199
200	184	168	152	136	135	119	103	87	71	55	39	23	7	216
217	201	185	169	153	137	121	120	104	88	72	56	40	24	8
9	218	202	186	170	154	138	122	106	105	89	73	57	41	25
26	10	219	203	187	171	155	139	123	107	91	90	74	58	42
43	27	11	220	204	188	172	156	140	124	108	92	76	75	59
60	44	28	12	221	205	189	173	157	141	125	109	93	77	61
62	46	45	29	13	222	206	190	174	158	142	126	110	94	78
79	63	47	31	30	14	223	207	191	175	159	143	127	111	95
96	80	64	48	32	16	15	224	208	192	176	160	144	128	112
113	97	81	65	49	33	17	1	225	209	193	177	161	145	129

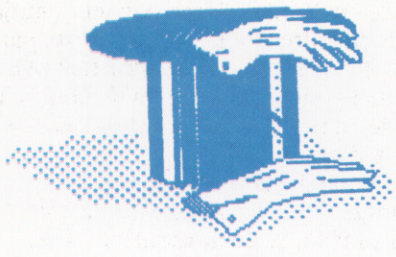
```
10 REM*************************************
15 REM***REMPLIR LES CARRES MAGIQUES*****
20 REM******SET-UP************************
30 M=19:DIM (A(M,M)
40 PRINT:PRINT"Carrés magiques"
50 PRINT : PRINT"Nombre de colonnes (1 à 9";
: INPUT S
60 IF S<0 OR S  INT(S)<>THEN  PRINT"ERREUR"
:GOTO 50
70 IF S>M THEN PRINT"ERREUR" :GOTO 50
80 IF S/2=INT(S/2) THEN PRINT"ERREUR - NOMBRE
IMPAIRS SEULEMENT" :GOTO 50
90 REM***REMPLIR LE CARRE*****************
100 X=INT(S/2)+1:Y=S:C=1
110 A(X,Y)=C
120 C=C+1:IF C>S*S THEN GOTO 200
130 X=X+1:IF X>S THEN X=1
140 Y=Y+1:IF Y>S THEN Y=1
150 IF A(X,Y)<>0 THEN X=X-2:Y=Y-1
160 IF Y=0 THEN Y=S
170 IF X=0 THEN X=S
180 IF X=-1 THEN X=S-1
190 GOTO 110
200 REM***                ****************
210 PRINT:PRINT
220 FOR Y=1 TO S:FOR X=1 TO S
230 A=A(X,Y):GOSUB 380:PRINT" ";A$;" ";
240 NEXT X:PRINT:NEXT Y

250 REM***VERIFIER COLONNES ET LIGNES****
260 F=0
270 FOR Y=1 TO S:T=0
280 FOR X=1 TO S:T=T+A(X,Y):NEXT X
290 IF F=0 THEN U=T:F=1
300 IF T<>U THEN PRINT"ERREUR - LES LIGNES
&";Y;"NE CORRESPONDENT PAS" :STOP
310 U=T:NEXT Y
320 FOR X=1 TO S:T=0
330 FOR Y=1 TO S:T=T+A(X,Y):NEXT Y
340 IF T<>U THEN PRINT"ERREUR - LA LIGNE 1 et
LA COLONNE " ;X;"NE CORRESPONDENT PAS" :STOP
350 U=T:NEXT X
360 PRINT:PRINT"TOUTES  LES  LIGNES  ET  TOUTES
LES  COLONNES  DONNENT  LE  MEME  RESULTAT  EN
ADDITION ";T
370 STOP
380 REM*****TRANSCRIPTION NOMBRE/CHAINE*****
390 A$=STR$(A)
400 IF LEN(A$)<3 THEN A$=" "+A$:GOTO 400
410 RETURN
```



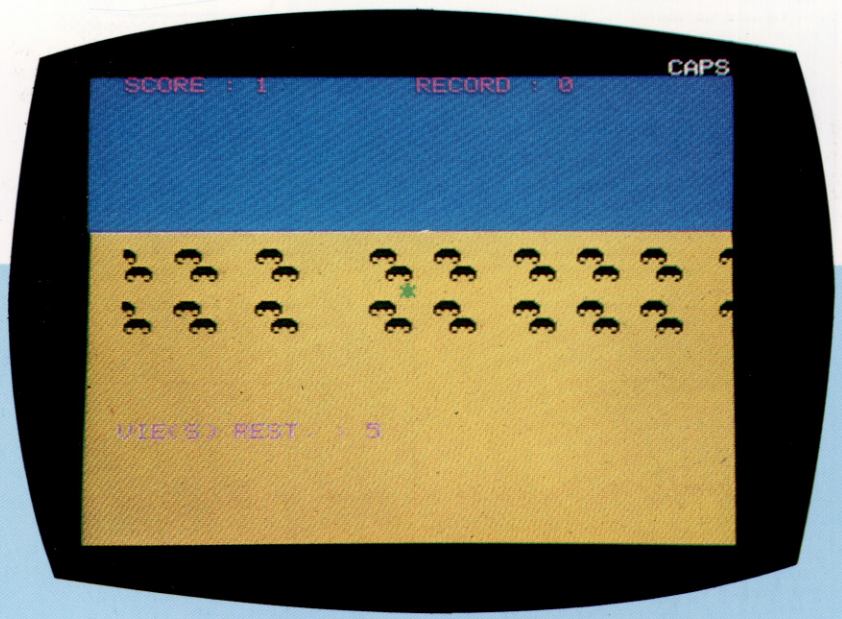
Variantes de basic

Ce programme a été écrit avec le basic Microsoft pour être applicable à beaucoup de micros de marché. Les utilisateurs du Spectrum devront faire figurer LET avant toutes instructions d'affectation. Le programme demande le nombre de colonnes (et donc de lignes), vérifie que ce nombre est bien positif, entier et impair. Puis il s'exécute et affiche le « Carré Magique ». Ensuite, à partir de la ligne 250, il vérifie ses propres résultats. Si cette dernière partie vous semble superflue, vous pouvez supprimer ces lignes 250-360.



Crabes

Les lois de la mer revues et corrigées pour de pures raisons informatiques. Pierre Monsaut est l'auteur de ce jeu écrit en basic pour l'ordinateur Atmos.



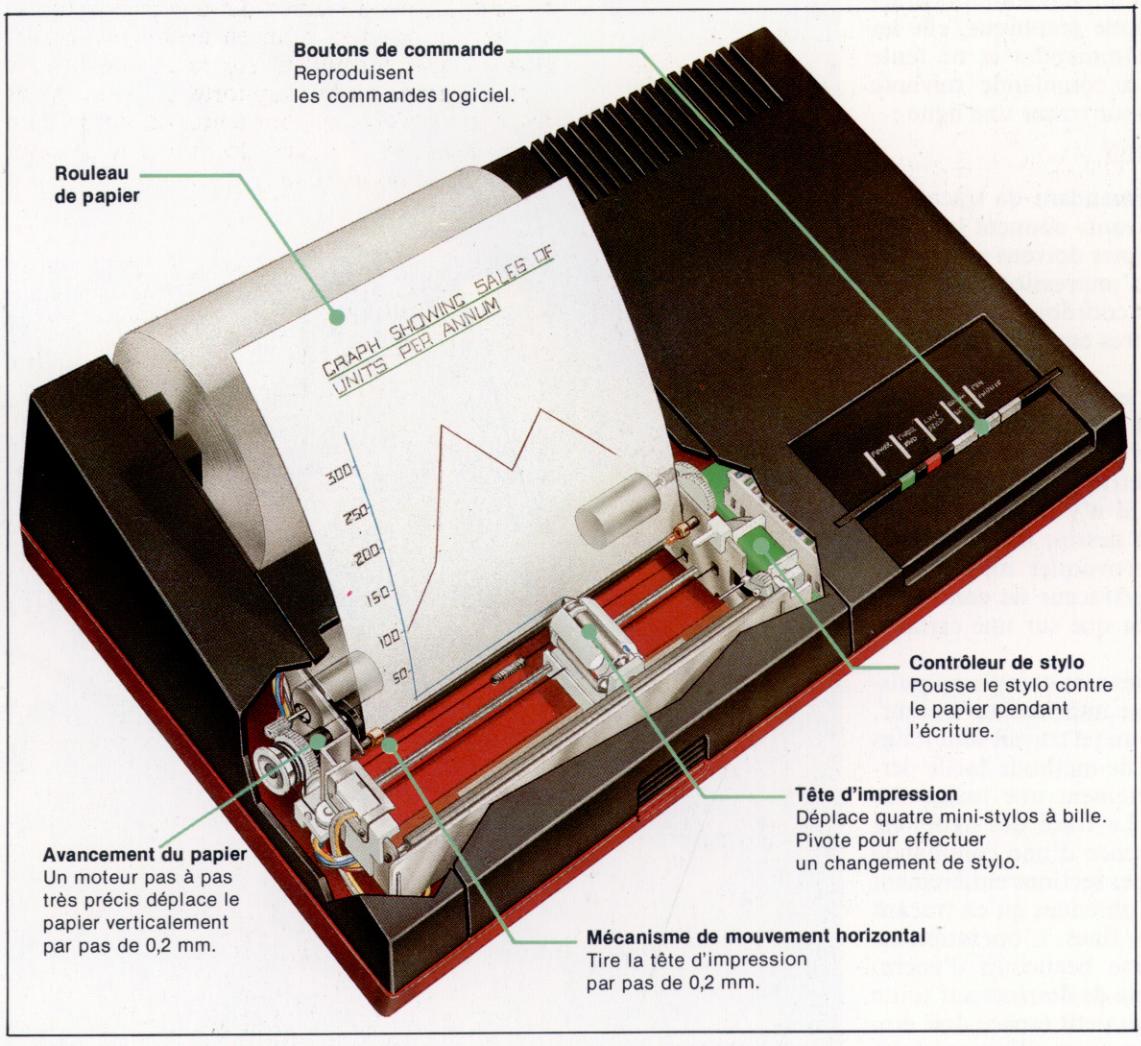
Vous devez maintenant aider une pauvre tortue de mer à regagner l'eau en évitant les crabes voraces qui patrouillent sur la plage. Chaque tortue amenée au but rapporte un point. Vous disposez de cinq vies pour tenter de marquer un score maximal. Utilisez les touches W pour avancer et Z pour reculer.

```
10 REM **********
20 REM * CRABES *
30 REM **********
35 POKE #26A,PEEK(#26A) OR 8
40 PRINT CHR$(17)
50 GOSUB 840
60 PLOT 2,20,"VIE(S) REST. :"+STR$(NP)
90 A$=RIGHT$(A$,1)+LEFT$(A$,37)
100 B$=RIGHT$(B$,37)+LEFT$(B$,1)
120 PLOT 2,X1,A$
140 PLOT 2,X2,B$
160 PLOT 2,X3,A$
180 PLOT 2,X4,B$
190 D$=KEY$
200 PY=PY+(D$="W")-(D$="Z")
210 IF PY>16 THEN PY=16
220 IF PY=8 THEN 360
230 C=SCRN(PX,PY)
240 IF C<>32 AND C<>91 THEN 550
260 PLOT PX,YP,N$
280 PLOT PX,PY,P$
300 YP=PY
320 T=T+1
330 IF T>500 THEN 660
340 GOTO 60
360 PLOT PX,YP,N$
380 PLOT PX,PY,P$
420 GOSUB 1600
460 PY=16
470 YP=PY
480 S=S+1
500 PLOT 2,0,"SCORE :"+STR$(S)
520 PLOT 20,0,"RECORD :"+STR$(R)
530 GOSUB 1110
540 GOTO 60
550 NP=NP-1
570 PLOT PX,YP,N$
580 PLOT PX,PY,"*"
600 GOSUB 1500
610 IF NP=0 THEN 660
620 PY=16
630 YP=PY
640 GOSUB 1110
650 GOTO 60
660 CLS
665 INK 0
670 IF S>R THEN R=S
680 IF T<500 THEN 720
690 PLOT 8,6,"** TEMPS ECOULE **"
720 PLOT 13,10,"SCORE :"+STR$(S)
740 PLOT 12,14,"RECORD :"+STR$(R)
760 PLOT 11,20,"UNE AUTRE ?"
770 REPEAT
780 D$=KEY$
790 UNTIL D$=""
795 REPEAT
800 D$=KEY$
805 UNTIL D$<>""
810 IF D$  "N" THEN 50
815 CLS
820 PAPER 7
825 INK 0
830 PRINT CHR$(17)
835 END
840 CLS
850 PAPER 3
860 INK 2
870 RESTORE
880 FOR I=0 TO 23
890 READ A
900 POKE 46808+I,A
910 NEXT I
920 P$=CHR$(91)
930 N$=CHR$(32)
940 A$=""
950 B$=""
960 S=0
970 NP=5
980 PX=19
990 PY=16
1000 YP=PY
1010 X1=10
1020 X2=11
1030 X3=13
1040 X4=14
1050 T=0
1060 FOR I=1 TO 38
1070 READ A
1080 A$=A$+CHR$(A)
1090 NEXT I
1100 B$=A$
1110 X=INT(RND(1)*36)+1
1120 A$=RIGHT$(A$,X)+LEFT$(A$,38-X)
1140 PLOT 1,20,CHR$(5)
1150 PLOT 1,X1,CHR$(0)
1160 PLOT 1,X2,CHR$(0)
1170 PLOT 1,X3,CHR$(0)
1180 PLOT 1,X4,CHR$(0)
1190 PLOT 0,0,CHR$(18)
1200 PLOT 1,0,CHR$(5)
1210 FOR X=0 TO X1-2
1220 PLOT 0,X,CHR$(20)
1230 NEXT X
1240 RETURN
1500 ZAP
1510 RETURN
1520 WAIT 300
1530 RETURN
1600 PING
1610 FOR PY=PY TO 0 STEP-1
1620 PLOT PX,YP,N$
1630 PLOT PX,PY,P$
1635 WAIT 30
1640 YP=PY
1650 NEXT PY
1660 PLOT PX,YP,N$
1670 WAIT 100
1680 RETURN
2000 DATA 12,45,63,30,30,63,45,0
2010 DATA 7,15,31,31,18,16,12,0
2020 DATA 56,60,62,62,18,2,12,0
2030 DATA 32,92,93,32,32,92,93,32,32,32
,92,93,32,32,32,32,32,92,93,32
2040 DATA 32,92,93,32,32,32,92,93,32,32
,92,93,32,32,92,93,32,32,32
```

© SYBEX 1984

Imprimante/traceur

Les traceurs sont des appareils complexes qui servent à produire des graphiques couleur détaillés sur papier. Difficile de s'offrir de telles machines en raison de leurs prix élevés.



Imprimante/traceur
L'imprimante/traceur a une résolution d'environ 500 × 2000 points dans le mode traceur, et un choix de dimensions d'impression. La couleur du stylo est choisie en faisant pivoter la tête d'impression, et le tracé et l'impression impliquent le déplacement horizontal de la tête d'impression vers la gauche et vers la droite et le déroulement avant et arrière du papier. Le stylo peut être appuyé contre le papier où s'en éloigner, ce qui permet de le déplacer en créant une ligne ou en laissant le papier intact. Le mécanisme est assez précis, mais des mouvements répétés du papier et de la tête d'impression peuvent provoquer un léger décalage du tracé. (Cl. Steve Cross.)

Atari 1020, Commodore 1510 et Oric MCP-40 ne sont que quelques noms sous lesquels est vendu un traceur bon marché. La conception de cette unité a été réalisée par un fabricant japonais et adaptée aux exigences de chaque société.

En plus de son potentiel graphique, le traceur peut aussi produire du texte, d'où son nom imprimante/traceur. L'unité comprend une tête pivotante qui possède quatre stylos à bille. Pour tracer une ligne, la tête pivote sur elle-même pour sélectionner la couleur désirée (rouge, bleu, vert et noir sont les couleurs standards), et le stylo choisi est alors appuyé contre le papier et déplacé. Une ligne horizontale est tracée par le mouvement latéral de la tête; une ligne verticale est produite par le mouvement de haut en bas du papier. L'imprimante/traceur stocke les configurations des lettres et des autres caractères dans sa propre mémoire. Lorsqu'un signal est reçu en provenance de l'ordinateur, l'imprimante/traceur trouve simplement le caractère correspondant dans sa mémoire interne et le trace comme s'il s'agissait d'une configuration graphique. La qualité d'impression obtenue est très bonne — certainement meilleure que la plupart des imprimantes matricielles bon marché.

En mode texte, l'imprimante/traceur fonctionne exactement comme toute autre imprimante. Bien que le papier utilisé ne fasse que 115 mm de largeur, l'unité peut produire des documents de 40 ou de 80 caractères par ligne.

L'étroitesse du papier a pour effet qu'un texte de 80 caractères est plutôt serré, mais la définition des caractères est bonne. Le texte peut être imprimé dans l'une des quatre couleurs à une vitesse d'impression d'environ 10 caractères à la seconde. C'est lent, comparativement à la plupart des imprimantes matricielles, mais presque égal à une sortie sur imprimante à marguerite. Pour produire des graphiques, on doit faire passer l'imprimante/traceur en mode graphique afin qu'elle puisse tracer des lignes, des courbes et des grosses lettres. Si l'unité reçoit des caractères tout en étant en mode graphique, elle les interprète comme des commandes et ne tente pas de les imprimer. La commande suivante serait utilisée en BASIC pour tracer une ligne :

```
LPRINT "D 0,100,100,100,100,0,0,0"
```

Le D est l'instruction demandant de tracer des lignes et les nombres suivants donnent les positions par lesquelles les lignes doivent passer. En supposant que la tête d'impression soit positionnée initialement aux coordonnées (0,0), un carré sera dessiné. D'autres commandes ont un format similaire.

La largeur du papier est divisée en 480 pas horizontaux qui servent à positionner la tête d'impression. Le papier utilisé est fourni en rouleaux de plusieurs mètres de longueur. Cela peut porter à croire qu'il n'y a aucune limite quant à la hauteur d'un dessin, mais le glissement du papier peut provoquer un décalage des lignes; l'imprimante/traceur ne consentira donc à déplacer le papier que sur une certaine distance.

Bien que des graphiques très complexes puissent être produits par une imprimante/traceur, les programmes pilotant un tel travail sont longs à décrire — il n'y a pas de méthode facile permettant de copier directement une image sur l'écran de l'ordinateur. Le tracé des lignes est assez simple, mais l'absence d'une commande PAINT ou FILL signifie que des sections entièrement colorées ne peuvent être obtenues qu'en traçant de nombreuses lignes très fines. L'opération est très longue et consomme beaucoup d'encre. L'unité est aussi incapable de dessiner sur toute la largeur du papier — un petit espace doit être laissé de chaque côté. Cet appareil est spécialement conçu pour produire des graphiques et reconnaît une commande qui trace automatiquement les axes d'un graphique, en plaçant les unités requises aux intervalles choisis.

En mode texte, l'imprimante/traceur produit deux dimensions de caractères. Cependant, en utilisant le mode graphique, des caractères beaucoup plus grands peuvent être obtenus et l'orientation des caractères modifiés, ce qui permet d'imprimer des messages verticalement ou diagonalement. L'encre des stylos à bille s'épuise rapidement, et les pointes des stylos ont tendance à sécher s'ils sont laissés sur la tête d'impression. Lors de la mise sous tension de l'imprimante/traceur, tous les stylos sont testés automatiquement puisque l'imprimante/traceur dessine un petit carré de chaque couleur.

Chris Stevens

L'imprimante/traceur permet de réaliser un tracé haute définition, d'utiliser plusieurs dimensions et plusieurs couleurs d'impression, de modifier l'orientation des caractères et de produire de véritables jambages.

Interfaces
Ce tableau montre quelles unités peuvent être utilisées avec les micros populaires. Le seul problème sera de trouver le bon câble pour les relier. Il faut signaler que d'autres marques de micros commencent à adapter des interfaces pour utiliser ce matériel.

	Atari 1020	Commodore 1520	MCP-40	Oric MCP-40	Sharp MZ-80P5(K)	Tandy GGP-115
Atari 400/600/800	●					
BCC Micro			●	●		●
CGL/Sord M5			●	●		●
Colour Genie						●
Commodore Vic-20		●				
Commodore 64		●				
Dragon 32/64			●	●		●
Memotech MTX			●	●		●
Oric-1/Atmos			●	●		●
Sharp MZ-700					●	
Tandy TSR-80 Colour						●

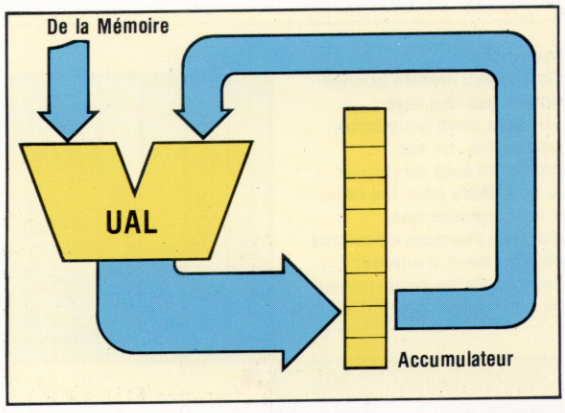
Fin logique

Dans ce dernier article sur la logique, nous étudions la logique de l'unité centrale (UC) et en particulier la fonction de l'unité arithmétique et logique (UAL). Un sujet particulièrement intéressant.

L'UAL a deux sortes de fonctions : arithmétiques avec l'addition, la soustraction et l'incrémentation (ajouter 1 à un nombre); logiques avec les opérateurs XOU (OU exclusif), IOU (OU Inclusif) et ET.

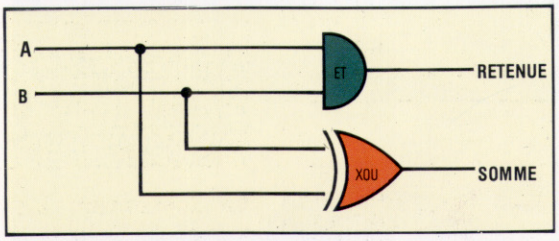
Certaines de ces fonctions, telles que l'addition, nécessitent deux opérandes (nombres sur lesquels porte l'opération). D'autres, telles que l'incrémentation, n'ont besoin que d'un seul opérande. Celui-ci est alors extrait d'un registre spécial de l'UC appelé accumulateur. Lorsqu'il faut deux opérandes, le deuxième provient de la mémoire principale. Les deux nombres sont alors traités par l'UAL et l'opération voulue est effectuée. Le résultat est placé dans l'accumulateur.

Le diagramme suivant montre le cheminement des données dans le composant de l'UAL :

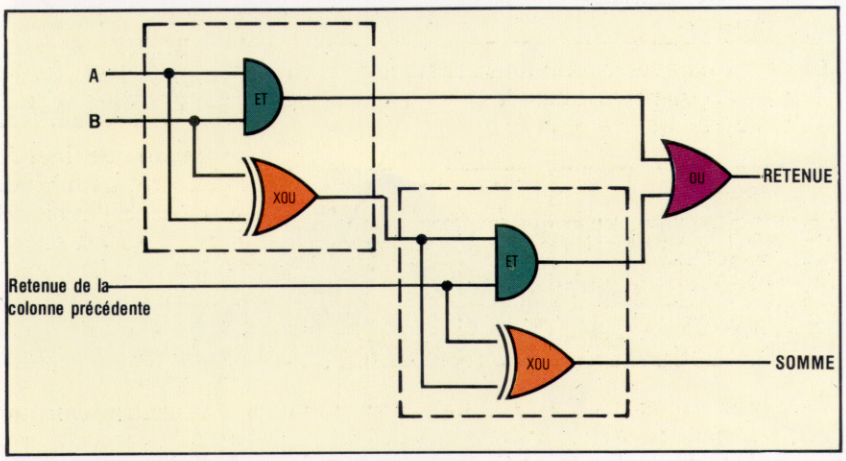


Les nombres présents dans l'accumulateur et dans la mémoire portent sur 8 bits. Ces bits constitutifs des nombres transitent en parallèle dans l'UAL et l'opération a lieu simultanément sur les 8 bits. Pour illustrer ce circuit, considérons seulement un des 8 bits. Nous allons créer un circuit convenant à l'exécution de 6 fonctions décrites plus haut. Nous appelerons le bit du premier opérande, A, et celui du second, B.

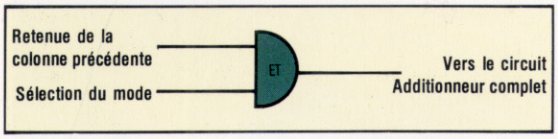
Le principe de l'élément de l'UAL que nous étudions est le circuit dit « additionneur complet ». Le circuit de ce type que nous avons déjà établi utilisait deux circuits demi-additionneurs, constitués de portes logiques ET, OU et NON. Ces circuits peuvent utiliser la porte XOU pour simplifier le circuit :



Les deux circuits demi-additionneurs s'assemblent pour former l'additionneur complet :



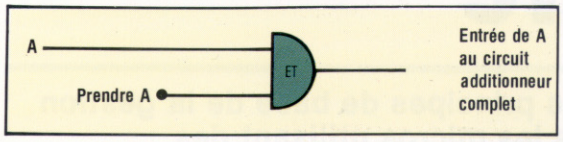
En rajoutant de petits circuits à l'additionneur complet, on découvrira comment les portes logiques s'adaptent aux autres fonctions de l'UAL. Nous allons disposer de signaux de commande, signaux dont le premier s'appelle « signal de sélection du mode ». L'entrée « retenue de la première colonne » est utilisée pour les opérations arithmétiques seulement. Le signal de sélection du mode active ou inhibe cette entrée au circuit grâce à la porte ET :

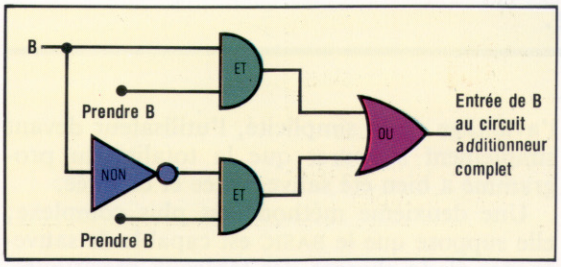


Pour les opérations arithmétiques où l'entrée « retenue » est nécessaire, le signal de sélection du mode sera mis à 1, donnant à la retenue l'accès à la porte ET. Dans le cas contraire, ce signal est mis à 0. De manière similaire, on peut ajouter des portes ET aux deux autres entrées au circuit. Cela permet de choisir le bit A ou le bit B, ou les deux à la fois.

Lors de l'opération de soustraction par addition du complément à 2, le complément à deux du nombre à soustraire devra être calculé. Cela suppose de changer tous les 0 en 1, et vice versa. Si nous voulons utiliser notre circuit additionneur pour les soustractions, nous allons devoir ajouter un circuit pour prendre la négation du bit B. Cela peut s'obtenir en affectant l'entrée B à la porte logique NON, le signal de sélection du mode pouvant être inclus au moyen d'une autre

porte ET. Le circuit final pour les entrées A et B et pour le signal de commande est le suivant :



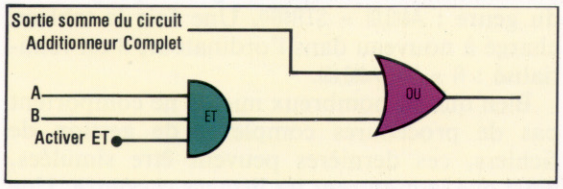


Réalisons les quatre fonctions arithmétiques en utilisant ces quatre signaux de commande. Cette table indique les combinaisons :

Fonction	Sel. mode	Prendre A	Prendre B	Prendre B
Addition	1	1	1	0
Soustraction	1	1	0	1
Incrémentation de A (mettre l'entrée pour la première retenue à 1)	1	1	0	0
Incrémentation de B	1	0	1	0

Nous pouvons mettre à 0 le signal de sélection du mode pour toutes les opérations. Cela signifie que la fonction logique XOU est obtenue à la sortie Somme en activant les modes Prendre A et Prendre B, et en inhibant le signal de sélection du mode.

La fonction ET ne peut s'obtenir directement. Elle suppose des entrées A et B distinctes ainsi qu'un signal de sélection ET.



La fonction OU, quant à elle, est obtenue en combinant les sorties XOU et ET sur la porte logique OU. La table de vérité suivante le montre :

A	B	Sortie	Commentaire
0	0	0	
0	1	1	Fonction XOU
1	0	1	
1	1	1	Fonction ET

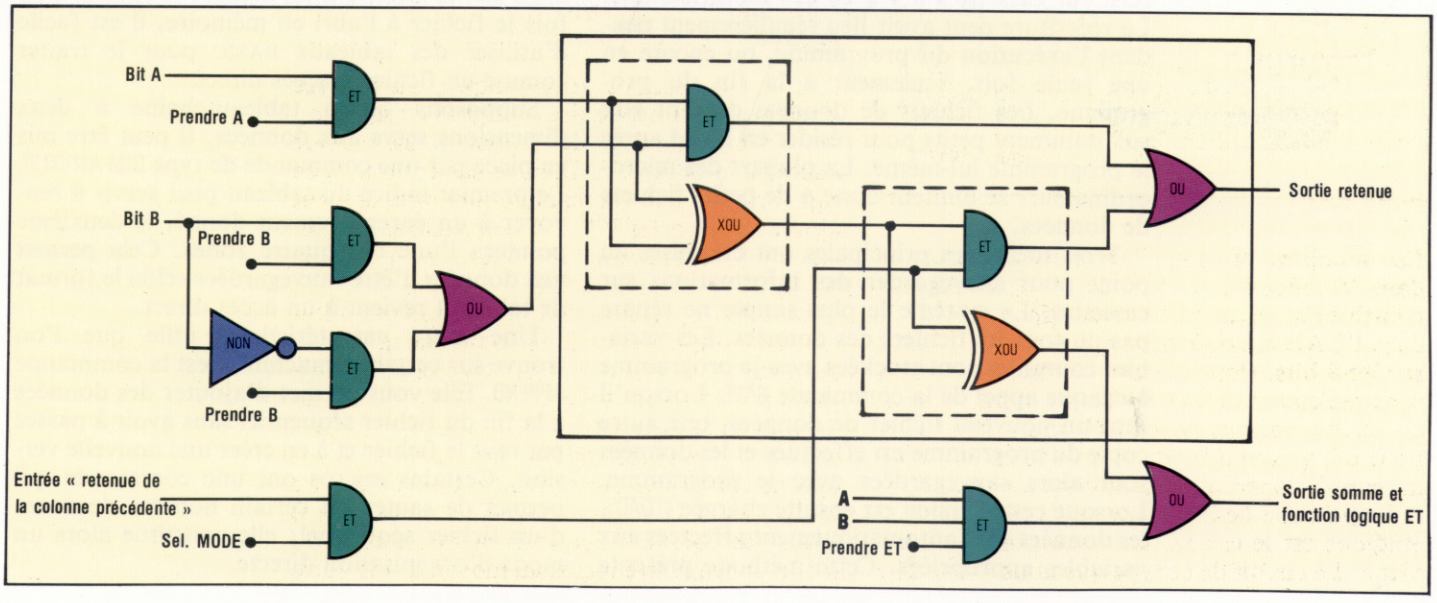
La table suivante indique comment s'obtiennent les fonctions logiques à partir de combinaisons de signaux de commande :

Fonction	Sel. mode	Prendre A	Prendre B	Prendre B	Prendre ET
XOU	0	1	1	0	0
ET	0	0	0	0	1
OU	0	1	1	0	1

Le diagramme final donné ci-dessous représente un élément à un bit de circuit de l'UAL. Il met en œuvre le circuit additionneur complet et les circuits logiques supplémentaires des signaux de commande. La totalité du circuit réunirait en parallèle 8 circuits de ce type. La sortie « retenue » de la 8e colonne sert de drapeau pour signaler une retenue pour le registre de statut du processeur.

Cet article conclut la série sur la logique. Nous l'avions commencé avec des notions abstraites telles que l'algèbre de Boole et les diagrammes de Venn. Nous avons ensuite abordé les circuits logiques simples et leurs résultats. Dernièrement, nous avons vu des circuits plus complexes qui concernent de près le fonctionnement interne de l'ordinateur.

Enfin, nous avons vu comment combiner plusieurs circuits afin de permettre au microprocesseur d'effectuer les opérations logiques et arithmétiques nécessaires. Chaque opération est commandée par une séquence de 0 et de 1 sur les lignes de commande de l'UAL. Ces séquences sont en fait des instructions en code machine sous leur véritable forme binaire.



Liz Dixon

Pour mémoire

Précédemment, nous avons traité des principes de base de la gestion de fichiers. Pour finir, nous abordons les micros utilisant des cassettes comme mémoire de masse.

Des micros différents utilisent des techniques de gestion de fichiers différentes. Aussi est-il souvent nécessaire que les programmes standard soient adaptés à une machine donnée. Il est donc important de connaître le rapport entre les caractéristiques et les commandes de votre machine, et les méthodes de gestion de fichiers en général. Voyons par exemple le stockage de fichiers de données sur un micro dont la mémoire de masse est sur cassette. Remarquons que les systèmes à cassettes, par leur nature même, ne peuvent gérer les fichiers à accès direct. L'accès aux données se fera selon l'ordre dans lequel elles ont été stockées, c'est-à-dire séquentiellement.

L'organisation séquentielle suppose de lire des informations d'un fichier, de les traiter et, ensuite, d'écrire les données mises à jour dans un deuxième fichier. Il faut donc qu'il y ait deux fichiers ouverts simultanément. Un lecteur à cassettes est incapable de se positionner successivement, directement et précisément sur des points distincts de la bande. Aussi ce système à deux fichiers est-il impossible pour la plupart des micros à cassettes. Les exceptions : les micros de la gamme Newbrain et Commodore Pet, qui disposent de deux ports cassettes, un pour la lecture et un pour l'écriture.

C'est pourquoi la plupart des micro-ordinateurs personnels se limitent à gérer un fichier séquentiel à la fois. D'où de sérieuses restrictions. Le fichier doit être chargé en mémoire avant traitement, pour être à nouveau écrit sur cassette s'il y a eu des modifications. La réécriture peut avoir lieu régulièrement pendant l'exécution du programme, ou encore en une seule fois, seulement à la fin du programme. Les fichiers de données doivent être suffisamment petits pour résider en RAM après le programme lui-même. La plupart des micro-ordinateurs se limitent donc à de petits fichiers de données.

Trois méthodes principales ont été mises au point pour sauvegarder des informations sur cassettes. Le système le plus simple ne sépare pas du tout les fichiers des données. Les variables courantes sont stockées avec le programme à chaque appel de la commande SAVE. Lorsqu'il faut un nouveau fichier de données, une autre copie du programme est effectuée et les données sont alors sauvegardées avec le programme. Lorsque cette version est ensuite chargée (LOAD), les données sont automatiquement affectées aux variables appropriées. Cette méthode présente l'avantage de la simplicité, l'utilisateur devant simplement s'assurer que la totalité du programme a bien été sauvegardée et chargée.

Une deuxième méthode est plus complexe; elle suppose que le BASIC est capable de sauvegarder et de charger des tableaux spécifiques. Sur l'Atmos d'Oric, par exemple, la commande STORE A$, "NOM" écrira le tableau A$ sur cassette, et la commande RECALL A$, "NOM" le chargera. L'intégralité du tableau (A$(1),A$(2), etc.), est chargée par SAVE, bien que les commandes STORE et RECALL n'indiquent pas la taille du tableau, cette dernière étant automatiquement donnée au début du programme lorsque le tableau est dimensionné par DIM.

Un des problèmes avec ce système est de garder trace du nombre d'entrées au tableau utilisées par le programme. Une solution est de stocker au tableau le total des enregistrements avant qu'ils ne soient sauvegardés. La plupart des machines acceptent l'indice 0, ce qui permet un total d'enregistrements de A$(0,0). Le compteur d'enregistrements peut être une variable numérique (R, dans notre programme), mais en présence d'un tableau de type chaîne de caractères, cela devra être une chaîne. La transcription s'obtient facilement en une ligne de programme du genre : A$(0,0) = STR$(R). Une fois le tableau chargé à nouveau dans l'ordinateur, R est réinitialisé : R = VAL(A$(0,0)).

Bien que de nombreux micros ne comportent pas de procédures complexes de gestion de fichiers, ces dernières peuvent être simulées, ainsi que le montrent les listages ci-contre. Une fois le fichier à l'abri en mémoire, il est facile d'utiliser des tableaux BASIC pour le traiter comme un fichier à accès direct.

Supposons qu'un tableau-chaîne à deux dimensions serve aux données. Il peut être mis en place par une commande de type DIM A$(100,3). Le premier indice du tableau peut servir à renvoyer à un enregistrement donné, le deuxième pointera l'une des quatre zones. Cela permet aux données d'être sauvegardées selon le format de table, et revient à un accès direct.

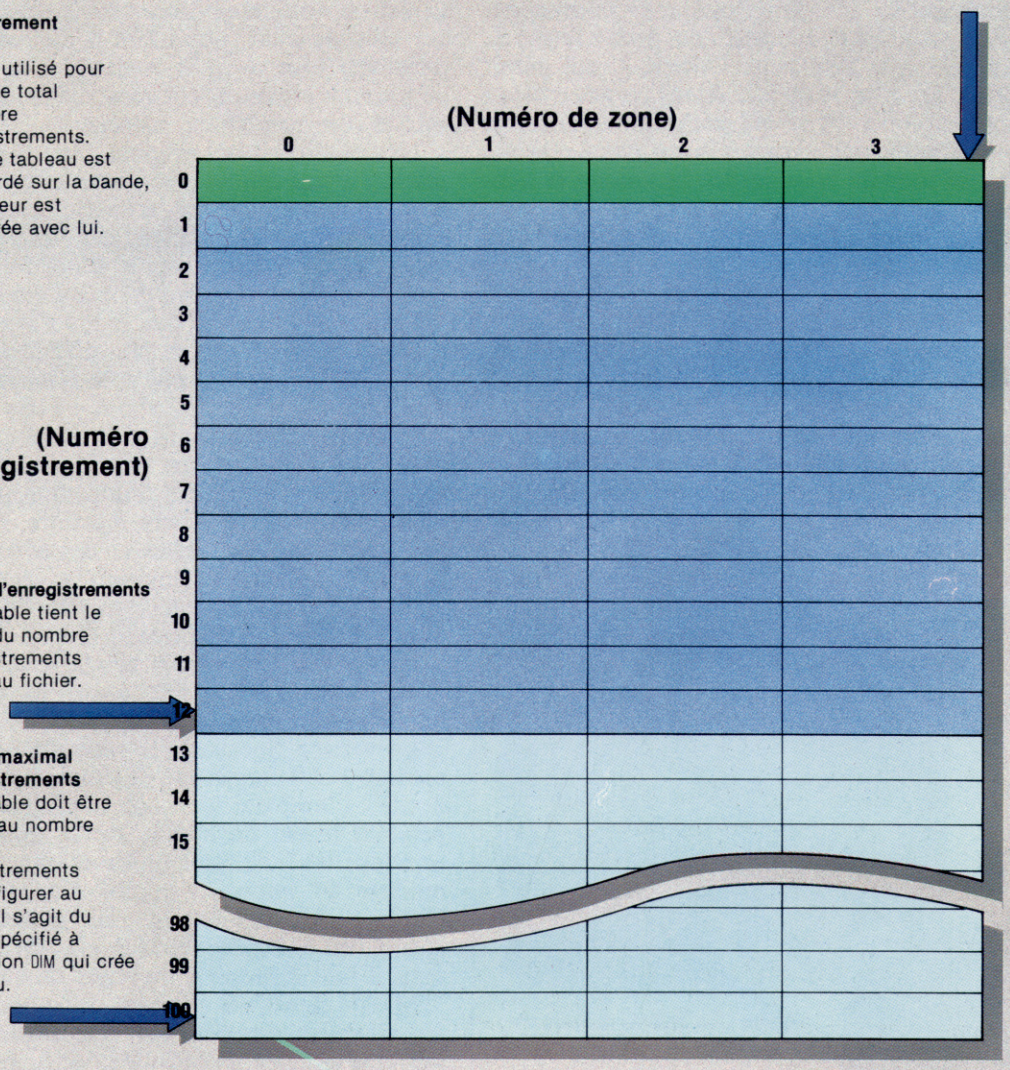
Une autre caractéristique utile que l'on trouve sur certaines machines, est la commande APPEND. Elle vous permet d'ajouter des données à la fin du fichier séquentiel sans avoir à passer par tout le fichier et à en créer une nouvelle version. Certains micros ont une commande qui permet de sauter un certain nombre de zones d'un fichier séquentiel; elle constitue alors un mode d'organisation directe.

Stocker des enregistrements et des zones dans un tableau basic

Un tableau-chaîne à deux dimensions peut permettre de simuler l'accès direct. Le premier indice sert à identifier un enregistrement, et le deuxième, les zones à l'intérieur de l'enregistrement.

Nombre de zones
Il est commode que la définition de ce mot soit dans une variable en début de programme. Cela facilite les modifications lorsque de nouvelles zones sont ajoutées par la suite, une seule instruction étant alors nécessaire.

Enregistrement en tête
A$(0,0) est utilisé pour stocker le total du nombre d'enregistrements. Quand le tableau est sauvegardé sur la bande, cette valeur est enregistrée avec lui.



Nombre d'enregistrements
Une variable tient le compte du nombre d'enregistrements utilisés au fichier.

Nombre maximal d'enregistrements
Une variable doit être affectée au nombre maximal d'enregistrements pouvant figurer au tableau. Il s'agit du nombre spécifié à l'instruction DIM qui crée le tableau.

Suppression d'un enregistrement au tableau
Ce segment de programme retire l'enregistrement numéro N du tableau, selon la méthode déjà exposée. Tous les enregistrements inférieurs à N sont déplacés d'une position vers le haut, ce qui a pour conséquence ultime de recouvrir en écriture les données stockées en N.

```
100  LET R = R + 1
110  IF R M THEN PRINT "TABLEAU PLEIN": RETURN
120  FOR I = R TO N + 1 STEP - 1
130  FOR J = 0 TO F
140  LET A$(I,J) = A$(I - 1,J)
150  NEXT J: NEXT I
170  LET A$(N,0) = N$: LET A$(N,1) = C$
180  LET A$(N,2) = D$: LET A$(N,3) = E$
190  RETURN
```

Ajout d'un enregistrement au tableau
Ce segment de programme insère un nouvel enregistrement en position N dans le fichier. Tous les enregistrements qui viennent après l'insertion sont décalés d'une position vers le bas pour ménager de la place au nouvel enregistrement sauvegardé en N$, C$, D$ et E$.

```
200  FOR I = N TO R - 1
210  FOR J = 0 TO F
220  LET A$(I,J) = A$(I + 1,J)
230  NEXT J: NEXT I
240  LET R = R - 1
250  RETURN
```

Kevin Jones

Chargement et sauvegarde d'un tableau d'enregistrements

Variables sauvegardées avec le programme (SAVE)
Le Spectrum sauvegarde ses variables en même temps que les programmes (il ne peut pas sauvegarder des tableaux seuls). Ce programme-type remplit un tableau avec les données, et l'instruction SAVE stocke le tableau avec le programme. Lorsque le programme est chargé à nouveau, il se positionne automatiquement sur la ligne 700. L'instruction RUN ne doit pas être utilisée.

Spectrum

```
100 REM***DEMO SPECTRUM***
200 DIM A$(100,20)
300 FOR K=1 TO 100
400 LET A$(K)="RecNo."+STR$(K)
500 NEXT K
600 STOP
700 PRINT "LE TABLEAU CONTIENT..."
800 FOR K=1 TO 300
900 PRINT A$(K),
1000 NEXT K
1100 STOP

SAVE "DEMOPROG" LINE 700

LOAD "DEMOPROG
```

Sauvegarde de tableaux avec noms
Le micro Atmos d'Oric comporte les commandes STORE et RECALL qui permettent de sauvegarder et de charger des tableaux d'une cassette.

Atmos de Oric

```
100 REM Sauvegarde un tableau sur cassette
110 A$(0,0)=STR$(R)
120 PRINT "Insérez la cassette, appuyez sur
PLAY et RECORD, et faites RETURN"
130 A$=KEY$:IF A$="" THEN 130
140 STORE A$, "FICHIER",5
150 PRINT "TERMINE"
160 RETURN

200 REM charger le tableau depuis la cassette
210 PRINT "Insérez la cassette, appuyez sur
PLAY, faites ensuite RETURN"
220 A$=KEY$:IF A$="" THEN 220
230 RECALL A$,"AFILE",5
240 R=VAL(A$(0,0))
250 PRINT "TERMINE"
260 RETURN
```

Emploi de fichiers séquentiels
Le BBC Micro est un des rares ordinateurs personnels à autoriser des fichiers séquentiels sur cassette. Ces deux sous-programmes sauvegardent et chargent le tableau en créant un fichier séquentiel.

BBC Micro

```
100 REM Sauvegarder un tableau sur cassette
105 PRINT "Insérez la cassette"
110 X=OPENOUT("FICHIER")
120 PRINTEX,R
130 FOR I= 1 TO R
140 FOR J= 1 TO F
150 PRINTEX,A$(I,J)
160 NEXT J: NEXT I
170 CLOSEEX
180 PRINT "TERMINE"
190 RETURN
200 REM Charger un tableau depuis la cassette
205 PRINT "Insérez la cassette, appuyez sur
PLAY, et faites ensuite RETURN"
210 IF INKEY(0)=-1 THEN 210
215 X=OPENIN("FICHIER")
220 INPUTEX,R
230 FOR I= 1 TO R
240 FOR J= 1 TO F
250 INPUTEX,A$(I,J)
260 NEXT J: NEXT I
270 CLOSEEX
280 PRINT"TERMINE"
290 RETURN
```

Sauts de puces

Bugabbo (ou Boogaboo) est un jeu très inattendu pour le Spectrum et le Commodore 64, dans lequel le joueur a le rôle d'une puce prisonnière au fond d'un puits.

Distribué par Quicksilva, Bugaboo comporte une séquence de générique très impressionnante, au cours de laquelle vous êtes informé de l'approche de la planète Cebella-7, qui paraît abriter des formes de vie. Après un atterrissage sans histoire, vous sautez gaiement au milieu de la végétation qui recouvre le sol de la planète, lorsque soudain vous perdez l'équilibre et tombez dans une profonde excavation, qui ouvre sur une immense grotte. Le but du jeu est de sortir de là.

Des saillies parmi les rochers de couleur bizarre fournissent des points d'appui au milieu d'un paysage qui évoque Jérôme Bosch : champignons multicolores, fleurs étranges, araignées grimpant le long des murs. Pour vous échapper, il vous faut sauter d'une corniche à l'autre, en évitant le dragon dévoreur de puces. La version destinée au Commodore 64 comporte même des fleurs carnivores.

L'écran joue un peu le rôle d'une fenêtre; à mesure que vous vous déplacez, de nouvelles portions de la caverne vous sont révélées. L'endroit couvre environ trois écrans de large sur cinq de haut, aussi faut-il du temps pour découvrir tout ce qu'il contient. Vous pouvez emprunter plusieurs chemins différents, et les saillies sont habilement placées.

Le scénario est très original, mais la programmation n'en exploite pas toutes les possibilités. Vous n'avez qu'une vie, et la séquence d'introduction revient chaque fois que vous la perdez — ce qui vous arrivera plus d'une fois, le dragon étant à peu près impossible à éviter.

Au fur et à mesure que le jeu progresse, les corniches sont de plus en plus difficiles à atteindre. Pourtant, il n'y a pas de nouveaux dangers aux niveaux supérieurs, ce qui est très regrettable : le programme aurait été bien plus passionnant s'il avait fallu lutter contre d'autres adversaires. Bien entendu, les écrans multiples occupent énormément d'espace mémoire, et laissent peu de place à des règles et des actions complexes; mais certains logiciels y sont tout de même parvenus : ainsi Jet Set Willy de Matthew Smith. La version destinée au Commodore 64 a un graphisme bien plus soigné que celle du Spectrum, mais ne fait guère usage de sa mémoire plus étendue.

La version Spectrum se joue au clavier; les touches 1 et 0 permettent de sauter à gauche et à droite. L'ampleur du saut dépend du temps pendant lequel on appuie sur la touche; plus ce temps est long, plus on saute loin. Le Commodore 64 impose l'emploi d'un manche à balai, et le bouton de mise à feu y joue le même rôle. C'est une méthode bien plus satisfaisante, mais il faut disposer d'un engin de qualité.

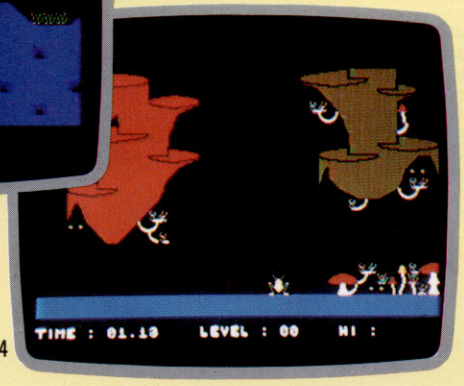
Malgré toutes ces critiques, Bugaboo reste un jeu vivement recommandé. Les contrôles sont des plus simples, et le joueur peut commencer immédiatement, sans avoir à se référer sans cesse aux instructions.

Sautez le pas

Bugaboo se signale avant tout par un graphisme remarquable. Il représente un puits profond semé de corniches sur lesquelles le joueur doit sauter afin de retrouver la liberté. La version Commodore est, d'un point de vue graphique, légèrement supérieure à celle du Spectrum.

Bugaboo (The Flea) sur le Spectrum

Bugaboo (The Flea) sur le Spectrum



Boogaboo (The Flea) sur le Commodore 64

Ian McKinnell

Bugaboo (The Flea) : pour Spectrum 48 K

Boogaboo (The Flea) : pour Commodore 64 + manche à balai

Éditeur : Quicksilva

Auteurs : Indescomp et Microbyte

Manche à balai : version Commodore seulement

Format : cassettes

Choix multiple

Avec les problèmes liés à la soustraction, nous abordons la programmation de la multiplication en langage machine, et une nouvelle classe d'opérations logiques.

Le Z80 et le 6502 comportent tous deux l'instruction SBC (soustraction avec retenue), mais leur mise en œuvre est différente. Sur le 6502, le drapeau de retenue est utilisé comme l'équivalent pour la soustraction de la retenue en addition. En langage d'assemblage Z80, SBC fonctionne exactement de la même manière que l'instruction ADC.

Supposons que l'on additionne $E4 à $5F en utilisant ADC (après avoir mis à zéro le drapeau de retenue). Le résultat dans l'accumulateur est $43, et le drapeau de retenue est mis, donc le résultat exact est $0143. Il y a eu débordement sur le drapeau de retenue car l'accumulateur ne peut contenir le résultat complet.

Supposons maintenant que sur le Z80, nous remettions à zéro le drapeau de retenue, et soustrayions $E4 de $5F : le résultat dans l'accumulateur est $7B, et le drapeau de retenue est mis. Si nous additionnons $7B à $E4 (après avoir remis encore le drapeau de retenue à zéro) nous trouvons comme résultat dans l'accumulateur $5F, et le drapeau de retenue est mis. Cela est tout à fait cohérent :

```
$5F - $E4 = $7B     Retenue
$5F = $E4 + $7B     Retenue
```

Si nous prenons l'état du drapeau de retenue comme indicateur d'un résultat négatif, nous pouvons alors interpréter $7B comme un complément à deux :

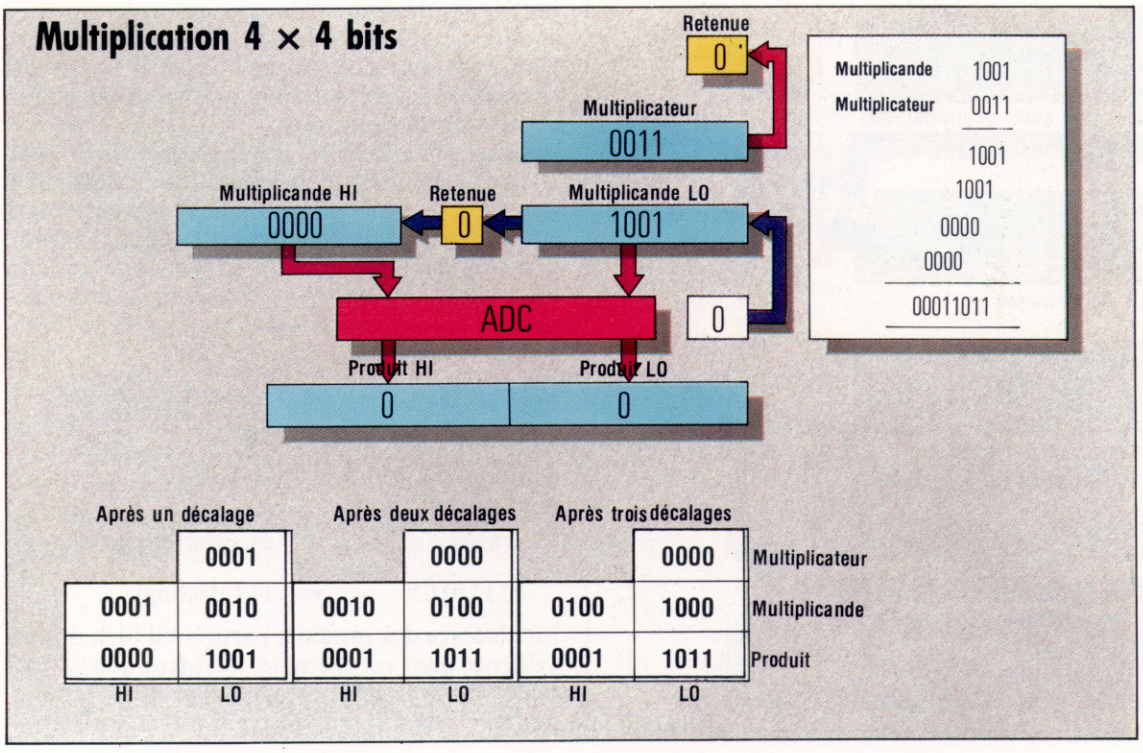
```
$7B en binaire                  = 01111011
Retrancher un                     - 1
                                ----------
Donne le complément à un          01111010
                          Non   ----------
Donne le complément à deux        10000101 = $85
```

Nous devrions alors nous attendre à ce que $5F - $E4 donne un résultat négatif -$85. Vérifions ce résultat en décimal :

```
 $5F =    95 décimal
-$E4 =   228 décimal
-----------------------
-$85 =  -133 décimal
-----------------------
```

On voit que tout cela est sensé. Supposons maintenant que la soustraction en question soit en fait une somme à deux octets : $375F - $21E4.

```
 HI  LO
 $37 5F   =  14175 décimal
-$21 E4   =  -8676 décimal
-------------------------
 $15 7B   =   5499 décimal
-------------------------
```



Temps de décalage
Cet exemple montre une multiplication de quatre bits, par souci de clarté — le nombre de bits n'affecte pas l'algorithme. Le produit est formé par l'addition de zéros ou de versions décalées du multiplicande, selon que le bit du multiplicateur est zéro ou un. Les bits du multiplicateur sont décalés vers la droite par le drapeau de retenue, alors que ceux du multiplicande le sont vers la gauche, de l'octet lo à l'octet hi par le drapeau de retenue.

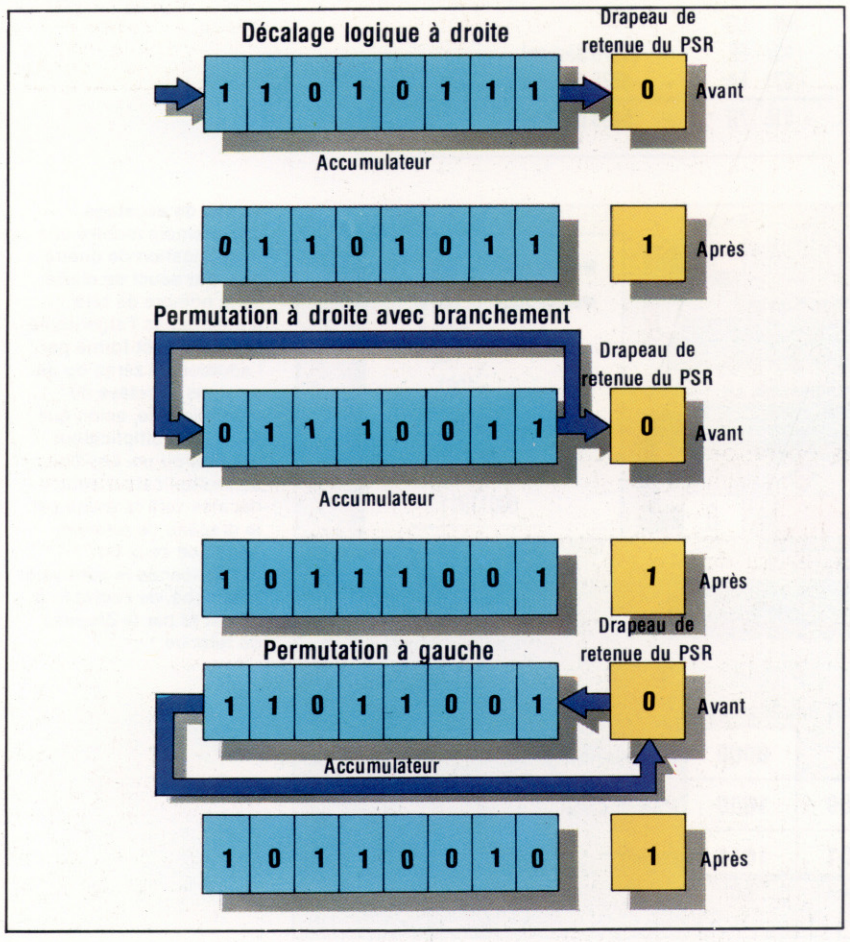
Nous savons que si nous effectuons la soustraction des octets lo, nous obtenons $7B avec retenue. Cette retenue est alors additionnée par l'instruction SBC à $21, ce qui donne $22, qui est ensuite soustrait de $37 pour donner $15. La réponse, $157B, peut être vérifiée par la version décimale.

L'arithmétique à deux octets sur le Z80 suit donc cette simple procédure :

1) Mettre à zéro le drapeau de retenue.
2) Soustraire les octets lo avec retenue.
3) Soustraire les octets hi avec retenue.

La version 6502 de cette séquence diffère sur le premier point — le drapeau de retenue doit être mis pour permettre un « emprunt » des octets hi aux octets lo. Si l'emprunt n'a pas lieu, la soustraction se passe normalement, et le drapeau de retenue reste mis pour la soustraction des octets hi. S'il y a débordement dans la soustraction des octets lo, le drapeau de retenue joue le rôle de « neuvième bit » de l'accumulateur. Cela assure qu'un résultat exact a été obtenu, et que le drapeau de retenue est ensuite remis à zéro. Lorsque les octets hi sont soustraits avec un drapeau de retenue à zéro, l'effet est le même que dans la soustraction des octets hi en Z80 avec le drapeau de retenue mis — le nombre à soustraire est décrémenté avant que la soustraction n'ait lieu. Les deux méthodes de soustraction ont leur équivalent dans les anciennes méthodes d'arithmétique où l'on « emprunte » ici pour « rembourser » là.

Travail de décalage
Les instructions de décalage et permutation sont utilisées d'abord pour examiner le contenu d'un registre bit par bit. A chaque décalage, le bit supérieur ou inférieur du registre est déplacé dans le drapeau de retenue du PSR; l'état de ce drapeau peut donc servir à une instruction de branchement pour déterminer le déroulement du contrôle de programme. Les instructions de permutation peuvent servir à conserver le contenu du registre, mais les instructions de décalage logique font entrer des zéros à mesure qu'elles font sortir des bits. Un décalage à gauche multiplie donc le contenu du registre par deux, et un décalage à droite le divise par deux.



Si l'on met à zéro le drapeau de retenue et soustrait $E4 de $5F, le résultat est $7A dans l'accumulateur, et le drapeau de retenue reste à zéro. Nous avons vu avec l'exemple du Z80 que le « bon » résultat est $7B avec le drapeau de retenue pour indiquer un nombre négatif. $7B est le complément à deux de la « vraie » réponse (−$85). Nous pouvons voir que $7A est le complément à un de ce nombre, et que l'état du drapeau de retenue est donc une sorte d'interrupteur sur l'état de l'accumulateur. C'est-à-dire qu'il est mis pour le complément à deux, et remis à zéro pour le complément à un.

Si nous faisons la soustraction sur le 6502 avec le drapeau de retenue mis, alors l'accumulateur contient $7B, et le drapeau de retenue est remis à zéro. S'il s'agit d'une soustraction à deux octets, le fait de remettre à zéro le drapeau de retenue assurera que le résultat de la soustraction des octets hi sera décrémenté, pour tenir compte de l'« emprunt » des octets lo.

Multiplication

Considérons la multiplication suivante :

174	multiplicande
× 209	multiplicateur
1566	1er produit partiel
000	2e produit partiel
+ 348	3e produit partiel
36366	produit final

Il n'est pas nécessaire de comprendre la notation positionnelle pour utiliser cette méthode, il suffit de suivre des procédures simples et de faire une multiplication à un chiffre. Le cœur de la méthode consiste à écrire chaque produit partiel décalé d'une place par rapport au précédent (les colonnes vides sont laissées en blanc). Une fois qu'on a accepté cela, il suffit de connaître les tables de multiplication pour former les produits partiels.

La combinaison des produits partiels décalés et de l'apprentissage machinal des tables, voilà pourquoi les longues multiplications décimales soit difficiles pour bien des gens. Dans la multiplication binaire, il n'y a qu'un réel produit, c'est 1 fois 1; tous les autres produits ont pour résultat zéro. Considérons cette longue multiplication binaire :

1101	=	13 décimal
× 1001	=	9 décimal
1101		1er produit partiel
0000		2e produit partiel
0000		3e produit partiel
1101		4e produit partiel
1110101	=	117 décimal

Le décalage des produits partiels est bien mis en évidence dans cet exemple, de même que la simplicité de la multiplication binaire. Un produit partiel est égal soit à zéro, soit au multiplicande

décalé, selon que le bit multiplicateur correspondant est un ou zéro. Cela évoque l'espèce de test que nous avons utilisé comme structure de contrôle en langage d'assemblage. Pour effectuer une multiplication binaire, examiner chaque bit du multiplicateur tour à tour, et ajouter zéro (si le bit est nul) ou le multiplicande décalé (si le bit est égal à un) au total.

Le test de l'état d'un bit particulier dans un octet peut être fait en utilisant l'instruction BIT sur les deux processeurs Z80 et 6502. Sur le Z80, cette instruction prend une adresse et un numéro de bit comme opérandes, et le drapeau de zéro est mis si ce bit particulier est nul, et enlevé si le bit est égal à un. Sur le 6502, l'opérande est une adresse. Le contenu de cette adresse est mis en relation avec l'accumulateur par ET, et le drapeau de zéro est mis ou non.

Ces instructions permettent une programmation subtile, mais aucune méthode ne convient particulièrement ici. Ce serait bien plus pratique si le bit en question pouvait agir comme le drapeau de retenue ou de zéro, de sorte que le programme se brancherait automatiquement selon l'état de chaque bit tour à tour. Inutile de dire que le jeu d'instructions des deux processeurs permettent cela grâce aux instructions de *décalage,* qui résolvent celui du multiplicande.

Il y a diverses instructions de décalage et de *permutation* dans les deux jeux d'instructions, bien que celles du Z80 soient plus complexes que celles du 6502. En général, leur effet est de décaler chaque bit d'un registre d'une position vers la droite ou la gauche. Elles diffèrent dans le détail pour le traitement du bit final du registre — un bit doit être décalé hors du registre à une extrémité tandis qu'un autre bit est décalé à l'intérieur à l'autre extrémité. Si le bit 7 sort du registre et y retourne en bit 0, alors l'opération est une *permutation gauche.* Si c'est le bit 0 qui devient le bit 7, alors c'est une *permutation droite.* Cela fait, le contenu du registre change d'ordre, mais aucune nouvelle valeur n'est introduite, et après huit de ces permutations circulaires le registre retrouve son état initial.

Si l'on n'utilise pas de permutations, il faut trouver une destination pour le bit sortant, et une source pour le bit rentrant. La plupart du temps, tous deux sont fournis par les divers drapeaux de condition du registre d'état du processeur (PSR), et en particulier le drapeau de retenue. En construisant un sous-programme de multiplication pour multiplier deux nombres à un octet, il faut décaler le multiplicande à gauche et le multiplicateur à droite. Les bits du multiplicande doivent être décalés vers l'octet hi du multiplicande, tandis que des zéros sont décalés vers les bits inoccupés. Les bits du multiplicateur doivent être décalés par un drapeau du PSR servant de test, mais leur destination, ainsi que l'état des bits multiplicateurs entrés par décalage, n'a pas d'importance à moins que nous ne devions conserver le contenu du multiplicateur. Tout ce qui compte au sujet du multiplicateur pendant la multiplication est de savoir si le bit sorti est un ou zéro.

Étant donné que le multiplicateur est stocké à l'adresse MPR, le multiplicande à MPDLO, et le produit à PRODLO et PRODHI, nous pouvons écrire ces sous-programmes de la manière suivante :

MULTIPLICATION DE HUIT BITS

6502			Z80		
	ORG	SC100		ORG	$D000
START	LDA	# $00	START	LD	BC,(MPR)
	STA	PRODLO		LD	B,$08
	STA	PRODHI		LD	DÈ,(MPDLO)
	STA	MPDHI		LD	D,$00
	LDX	# 8		LD	HL,$00
	CLC		LOOP0	SRL	C
LOOP0	ROR	MPR		JR	NC,CONT0
	BCC	CONT0		CALL	ADDIT
	JSR	ADDIT	CONT0	SLA	E
CONT0	ASL	MPDLO	ENDLP0	DJNZ	LOOP0
	ROL	MPDHI		LD	PRODLO
	DEX			RTS	
ENDLP0	BNE	LOOP0	MPR	DB	$E2
	RTS		MPDLO	DB	$7A
MPR	DB	$E2	MPDHI	DB	$00
MPDLO	DB	$7A	PRODLO	DW	$0000
MPDHI	DB	$00	ADDIT	ADD	HL,DE
PRODLO	DB	$00		RET	
PRODHI	DB	$00			
ADDIT	CLC				
	LDA	PRODLO			
	ADC	MPDLO			
	STA	PRODLO			
	LDA	PRODHI			
	ADC	MPDHI			
	STA	PRODHI			
	RTS				

Il ressort de cet exemple que la programmation du Z80 est beaucoup plus facile du fait de ses registres à 16 bits et des instructions associées. En particulier, comparez le sous-programme ADDIT dans les deux versions. En 6502 on utilise ROL pour permuter le multiplicateur vers la droite, et ASL et ROL pour décaler le multiplicande vers la gauche en passant de MPDLO à MPDHI. La boucle est contrôlée par le registre B qui sert de compteur. Notez que l'instruction ADD non seulement comporte une arithmétique de registre à 16 bits, mais également n'est pas affectée par le drapeau de retenue — contrairement à ADC.

Dans le prochain cours, nous discuterons des méthodes de division, et considérerons différents moyens de contrôler l'affichage à l'écran. Cela complètera la base théorique du cours.

Exercice 15

1. Écrire un sous-programme de multiplication utilisant un multiplicande à 16 bits et un multiplicateur à 8 bits de votre choix.
2. La multiplication n'est qu'une répétition d'additions : écrire un sous-programme de multiplication de 8 bits par 8 bits qui n'utilise pas les instructions de décalage ni de permutation.

Élégance à l'italienne

La firme Olivetti, une multinationale italienne, est parvenue à se créer une réputation de créatrice de machines à la fois élégantes et solides, et joue un rôle important en ce domaine.

Le souci du design
L'ordinateur de gestion Olivetti M20 est un 16-bits apparu en 1981. Il est construit autour d'un microprocesseur Z8000 de Zilog et a recours au système d'exploitation PCOS mis au point par la firme elle-même. Une version compatible avec l'IMB PC est sortie récemment.

En 1908, Camillo Olivetti fonda à Ivrea, une petite ville de l'Italie du Nord, une entreprise employant vingt personnes, et commença la production du premier modèle, machine à écrire de la compagnie, la M1. A cette époque, l'économie italienne était encore largement dominée par l'agriculture, et l'industrie lourde, qui avait assuré l'essor de l'Allemagne, de la Grande-Bretagne ou de la France, n'y existait guère. Pourtant la production d'Olivetti connut une croissance continue, passant de quatre machines par jour en 1914 à cinquante en 1929.

Dans les années trente, Adriano Olivetti, le fils du fondateur, procéda à une profonde réorganisation, et embaucha comme cadres des élèves de l'école du soir de la compagnie, fondée en 1924. Une politique sociale qui évoque assez « l'emploi à vie » des Japonais fut également mise en œuvre : logements, avantages sociaux. Alors même que l'économie mondiale s'efforçait de résister à la dépression économique de l'avant-guerre, Olivetti poursuivit son essor : en 1933, la compagnie avait déjà vendu quinze millions de produits de bureau. Elle sortit en 1937 son premier téléscripteur, et en 1940 sa première calculatrice.

La guerre marqua bien entendu un temps d'arrêt, mais après 1945 la compagnie entreprit de s'étendre sur de nouveaux marchés; c'est ainsi qu'en 1948 elle s'installa en Grande-Bretagne. Son succès reposait sur des produits de qualité, au dessin particulièrement élégant; même les responsables d'IBM durent admettre que ces machines « s'harmonisaient entre elles comme les pièces d'un puzzle ».

Au cours des vingt ans qui suivirent, la firme aborda le domaine des ordinateurs, sortant une calculatrice numérique dès 1955, tandis que son premier gros système, Elea, était mis au point quelques années plus tard.

Olivetti entreprit de se diversifier davantage, renonçant à l'équipement de bureau mécanique pour passer au matériel électronique, un domaine où elle joue un rôle pilote. Une nouvelle série de mini-ordinateurs apparut, avec des terminaux destinés aux banques et des systèmes de communication.

Le catalogue actuel est particulièrement vaste — plus d'un millier de produits — et la compagnie consacre des sommes importantes à la création de logiciels pour ses machines. En 1982, Olivetti était devenu le second constructeur d'ordinateurs européen (dépassé seulement par IBM), grâce à l'ordinateur portable M10 et au M20, destiné à la gestion : tous deux se vendent particulièrement bien.

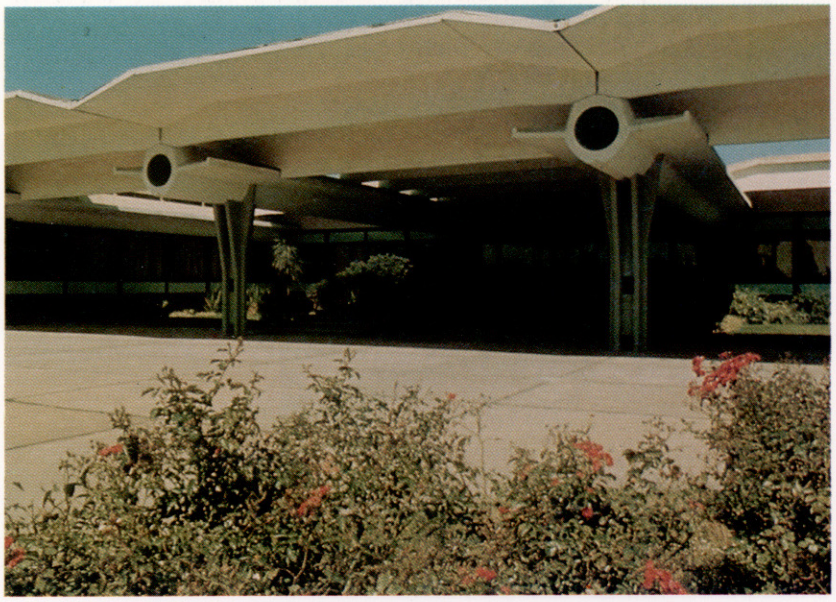
Le M10 ne pèse que mille sept cents grammes; il dispose d'un écran à huit lignes de quarante caractères. L'appareil est alimenté par piles, et a une RAM de 8 K, extensible à 64 K. Le M20 est un ordinateur de gestion 16 bits, construit autour d'un microprocesseur Z8001, qui n'a pas connu un grand succès auprès des autres constructeurs. Le M20 est également équipé d'un 8086, ce qui autorise une certaine compatibilité avec le CP/M-86 et le MS-DOS.

Olivetti prévoit aussi de lancer un compatible IBM, et annonce qu'il sera moins cher que le PC. Appelé M24, il sera doté d'un microprocesseur 8086-2 et, en option, d'une carte Z8001, afin de le rendre compatible avec le M20. Cela signifie que la firme a été contrainte d'abandonner son propre système d'exploitation, le PCOS.

Olivetti a récemment signé un contrat avec ATT (la plus grosse compagnie de télécommunications du monde), afin de collaborer à un projet de développement du système d'exploitation Unix. Il n'y a aucun doute qu'à l'avenir le réseau de vente de la firme et ses équipes de recherche et de développement lui permettront de maintenir sa réputation : des produits de grande classe, remarquablement fabriqués.

Mais remarquons que cette coopération apporte une preuve supplémentaire de l'échec d'une future information européenne.

L'élégance avant tout
Olivetti est célèbre pour le dessin très élaboré de ses produits. Ce souci d'élégance marque aussi l'architecture de ses bureaux. Celui-ci fut construit en 1959 et beaucoup de ses éléments ont été repris par d'autres architectes.



PROGRAMME N° 23

PROGRAMME DE TRI (suite)

Nous avons vu précédemment un programme permettant de rechercher simplement la note la plus élevée de la table VA( ). Allons plus loin cette fois et sophistiquons le programme pour déterminer un véritable tri de la table VA.
Il convient, auparavant, de faire une remarque. Les traitements effectués jusqu'à présent (TOTALISATEUR, MOYENNE, RECHERCHE DE LA MEILLEURE NOTE) ne nécessitent pas obligatoirement l'utilisation d'une table. Par contre, dès que l'on attaque un tri, l'utilisation des tables est obligatoire.

Trions « VA( ) »

Comparons le 2e élément au 1er élément :
— si ce 2e élément est plus grand que le 1er élément, nous les laissons dans l'ordre;
— si le 2e élément est plus petit que le 1er élément, nous inversons leur place pour qu'ils soient dans l'ordre.

Ligne 120

```
IF VA(I + 1) < VA(I) THEN Z = VA(I) : VA(I) = VA
(I + 1) : VA (I + 1) = Z:TM = 1
```

L'inversion de VA(1) et de VA(I+1) se fait en utilisant la variable Z.
Ensuite, nous comparons de la même façon le 3e élément de la table au 2e élément (en faisant I = I + 1) et nous renouvelons l'opération précédente. Inversion s'ils ne sont pas dans l'ordre. Stand-by s'ils sont dans l'ordre. Après avoir comparé tous les éléments adjacents de la table, le plus grand des éléments (donc la meilleure note) est positionné en fin de table.
Remarque. — Il n'y a que I - 1 comparaisons pour I éléments. Cependant, le tri n'est pas obligatoirement terminé.
Pour vérifier, il faut tester le témoin d'inversion TM. S'il n'y a pas d'inversion au cours de la lecture de la table, c'est que

VA(1) < VA(2) < VA(3) < VA(4) < VA(5)

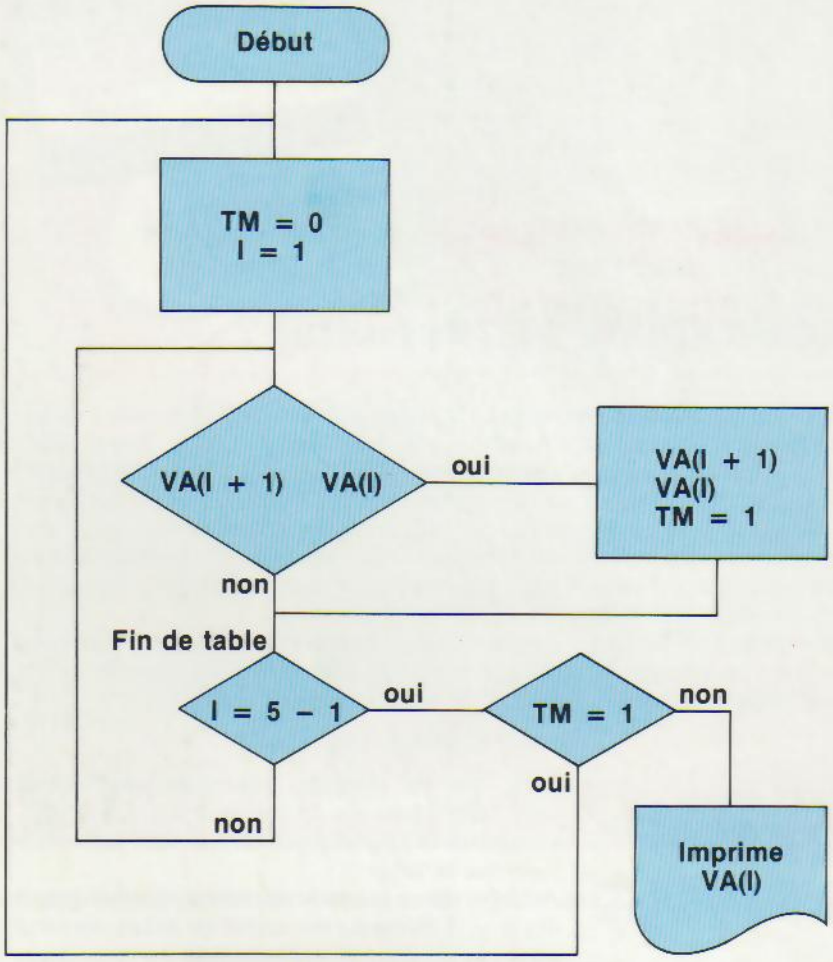
donc que les éléments sont bien classés par ordre croissant. En revanche, s'il y a eu inversion (dans la majorité des cas), il faut explorer à nouveau la table.
A la fin de ce second traitement, le plus grand des $n - 1$ éléments est arrivé en avant-dernière position et donc n explorations au maximum sont nécessaires au tri complet de la table.

16	12	12
12	10	10
10	13	7
13	7	13
7	16	16
	après le 1er passage	après le 2e passage

```
5   REM  BOUCLE DE SAISIE DES VALEURS
10  FOR I = 1 TO 5
20  INPUT "VALEUR ?:";VA(I)
30  NEXT I
95  REM  INITIALISATION DU TEMOIN D'INVERSION A 0
100 TM = 0
105 REM  BOUCLE DE COMPARAISON ET D'INVERSION EVENTUELLE
106 REM  SI L'INVERSION A LIEU LE TEMOIN TM PREND LA VALEUR 1
110 FOR I = 1 TO 5 - 1
120 IF VA(I + 1) < VA(I) THEN Z = VA(I):VA(I) = VA(I + 1):VA
    (I + 1) = Z:TM = 1
130 NEXT I
135 REM  TEST DE TM
140 IF TM = 1 THEN  GOTO 100
195 REM  EDITION DE LA TABLE TRIEE
200 FOR I = 1 TO 5
210 PRINT VA(I)
220 NEXT I
```

```
VALEUR ?:8
VALEUR ?:0
VALEUR ?:18
VALEUR ?:2
VALEUR ?:15
0
2
8
15
18
VALEUR ?:16
VALEUR ?:12
VALEUR ?:10
VALEUR ?:13
VALEUR ?:7
7
10
12
13
16
```

ORGANIGRAMME



On peut essayer de mieux suivre le traitement du tri en écrivant en 137 :

```
FOR K = 1 TO 5 : ? VA(K) : NEXT K : ?
```

pour faire imprimer la table après chaque investigation de la table. On obtient alors le programme suivant :

```
5   REM  BOUCLE DE SAISIE DES VALEURS
10  FOR I = 1 TO 5
20  INPUT "VALEUR ?:";VA(I)
30  NEXT I
95  REM  INITIALISATION DU TEMOIN D'INVERSION A Ø
100 TM = Ø
105 REM  BOUCLE DE COMPARAISON ET D'INVERSION EVENTUELLE
106 REM  SI L'INVERSION A LIEU LE TEMOIN TM PREND LA VALEUR 1
110 FOR I = 1 TO 5 - 1
120 IF VA(I + 1) < VA(I) THEN Z = VA(I):VA(I) = VA(I + 1):VA
    (I + 1) = 1
130 NEXT I
135 REM  TEXT DE TM
137 FOR K = 1 TO 5: PRINT VA(K): NEXT K: PRINT
140 IF TM = 1 THEN  GOTO 100
195 REM  EDITION DE LA TABLE TRIEE
200 FOR I = 1 TO 5
210 PRINT VA(I)
220 NEXT I
```

```
ÜRUN
VALEUR ?:16
VALEUR ?:12
VALEUR ?:10
VALEUR ?:13
VALEUR ?:7
 12
 10
 13
 7
 16
 10
 12
 7
 13
 16
 12
 7
 12
 13
 16
 7
 10
 12
 13
 16
 7
 10
 12
 13
 16
```

1er passage, le plus grand est en fin de table

2e passage, 13 vient en avant-dernière position

3e passage

etc.

table tirée définitive